

| | | |
|--|--|---|
|  PARIS PANTHÉON-ASSAS UNIVERSITÉ | L3-APP Rapport d'activité |  #RévélateurDeTalents |
|--|--|---|

Apprenti

Nom: **Logan**

Prénom: **TANN**

Parcours: **LSI**

Titre du rapport

Développement d'un intranet sur l'écosystème Salesforce

Entreprise

Nom: **Edifixio**

Adresse: *80 quai voltaire, 95 870 Bezons*

Soutenance

Date: _____

Heure: _____

Composition du jury :

- Responsable Efrei : _____
- Responsable en entreprise : _____
- Responsable CFA : _____

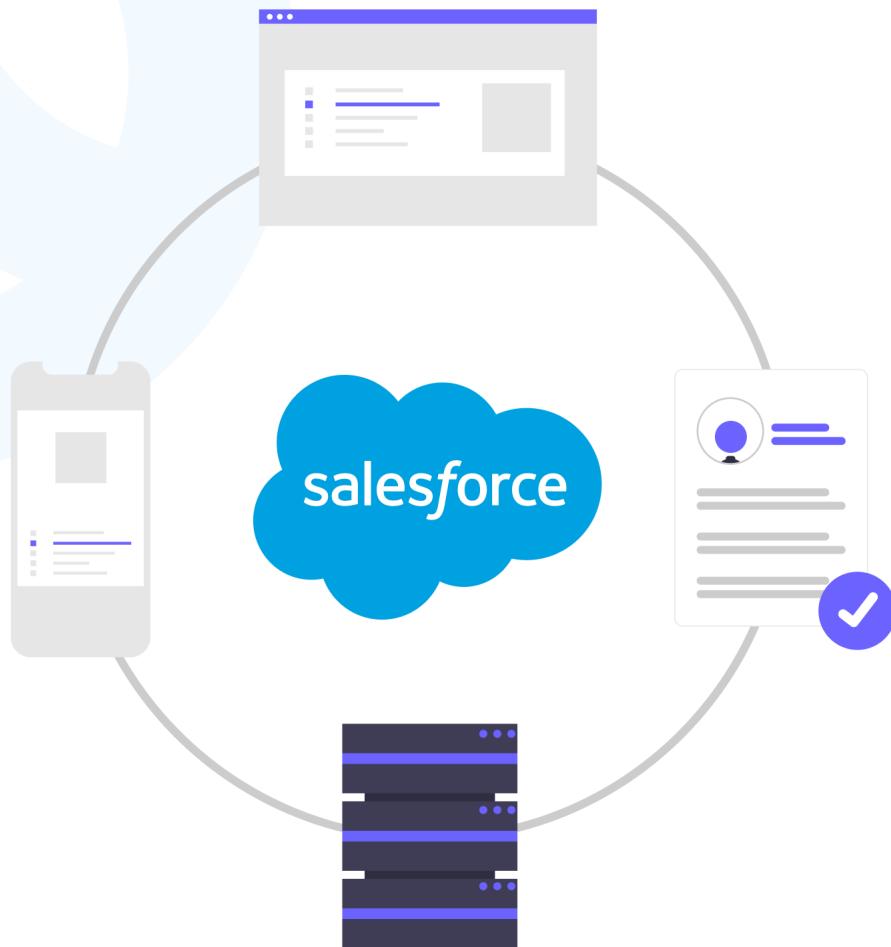
Confidentialité du rapport d'activité

- Rapport confidentiel: OUI / NON
- L'entreprise autorise la diffusion du rapport d'activité :
 - En version numérique sur le livret électronique : OUI / NON
 - En version papier : OUI / NON

Cachet de l'entreprise / Signature du responsable :

Mots clés

Développement full-stack, Ecosystème Salesforce.com, Programmation orientée objet, Programmation web, Base de données orientée objet, Architecture logicielle en couches, Logiciel de gestion relation client (CRM), Intranet, Langages de programmation : Apex - Javascript - SOQL



Rapport d'apprentissage 2022 - 2023

Développement d'un intranet sur l'écosystème Salesforce

Apprenti-ingénieur
Logane TANN
L3-App LSI (promo 2025)

Tuteur enseignant
Salim BOUZITOUNA
EFREI Paris

Maître d'apprentissage
Nicolas VERGER
Edifixio



ABSTRACT

As part of my software engineering apprenticeship at Edifixio, a French services company, I became a member of the development team responsible for internal projects.

Edifixio's information system comprises ten applications built on the proprietary Salesforce platform. From October 2022, I was responsible for four of the 10 while simultaneously gaining expertise in Salesforce development. In addition, I actively contributed to team building by leading workshops and creating two side-projects for the HR Team.

These missions were assigned as a continuity to the previous five-month internship experience during my DUT (between April and August 2022). However, in the last few months, Edifixio experienced many strategic changes, which had a direct impact on my current and future missions as part of the internal team.

Remerciements

Je voudrais tout d'abord remercier l'équipe pédagogique de l'EFREI, qui m'ont accompagné dans un cursus riche en expériences. Je remercie en particulier M. Salim Bouzitouna, pour la qualité de son accompagnement en tant que professeur référent ainsi que son incroyable réactivité à mes sollicitations.

Je tiens également à remercier mon responsable technique, M. Gautier Maufoy, ainsi que mon responsable fonctionnel, M. Matthieu Cuenin. Leur professionnalisme et leur disponibilité m'ont permis de bénéficier d'une formation en apprentissage très riche et correspondant à la réalité du métier. Sans oublier mon maître d'apprentissage, M. Nicolas Verger. Sa bonne humeur et son implication en tant que manager hiérarchique ont été très appréciées.

À ceci s'ajoute les camarades et alumnis de l'IUT d'Université Paris-Cité et de l'EFREI Paris qui ont travaillé avec nous dans les projets internes. En particulier Antoine Banha (L3-App LSI), Daniel Aguiar (M1-App LSI) et Arsène Lapostolet (alumni), auquel nous avons pu partager en commun une passion du code de qualité et des expériences variées en termes de technologies.

Table des matières

| | |
|---|----|
| Remerciements | 3 |
| Introduction | 5 |
| I) Présentation de l'environnement professionnel | 6 |
| I. A) L'entreprise Edifixio | 7 |
| I. B) Projet de fusion avec la société mère, « Eviden » | 7 |
| I. C) Partenariats : Expertises et clients | 8 |
| I. D) Le pôle Digital Platform | 8 |
| I. E) Organigramme | 8 |
| I. F) Vie en entreprise | 9 |
| II) Écosystème technique | 13 |
| II. A) L'écosystème Salesforce | 14 |
| II. B) Le <i>Salesforce Interne</i> , système d'informations d'Edifixio | 15 |
| II. C) Organisation du travail à réaliser | 16 |
| II. D) Comment développe-t-on sur Salesforce ? | 17 |
| II. E) Architecture logicielle | 20 |
| III) Détail des missions | 22 |
| III. A) Mon travail en tant qu'apprenti-ingénieur | 23 |
| III. B) L'outil de comptabilité (<i>Achats</i>) | 23 |
| III. C) Les outils de ressources humaines (<i>Tock</i> et <i>TRH</i>) | 25 |
| III. D) L'outil de gestion de projets (<i>Staffing</i>) | 26 |
| IV) Bilan | 32 |
| IV. A) Récapitulatif | 33 |
| IV. B) Difficultés rencontrées | 33 |
| IV. C) Grille eCF3 renseignée et datée | 33 |
| IV. D) Perspectives d'évolution | 34 |
| Conclusion | 35 |
| Table des figures | 36 |
| Bibliographie | 37 |
| [A] Annexes | 38 |
| [A1] Localisation des locaux d'Edifixio | 39 |
| [A2] Exemple de revue de qualité de code | 39 |
| [A3] Exemple d'un composant LWC | 40 |
| [A4] Support de la langue anglaise | 40 |
| [A5] Tests unitaires, tests sur Salesforce | 41 |
| [A6] Exemple de documentation technique d'un composant front-end | 42 |
| [A7] Gestion de projet en « Kanban » et versionnage de code | 43 |
| [A8] Architecture logicielle de l'application Validation Center | 44 |
| [A9] Exemple du réseau social d'entreprise, Chatter | 45 |
| Historique des révisions | 45 |

Introduction

Le 4 avril 2022, j'ai rejoint Edifixio dans le cadre d'un stage de fin d'études DUT, en tant que développeur full-stack spécialisé sur l'écosystème Salesforce. Par la suite, j'ai eu l'opportunité de prolonger mon expérience pendant trois années supplémentaires en réalisant une formation d'ingénierie logicielle en apprentissage à l'EFREI.

Durant une période de huit mois, j'ai pu contribuer à l'évolution d'une application de comptabilité appelée « Achat », et depuis février, j'ai également été impliqué dans la création d'un outil de planification de projets appelé « Staffing ».

Le système d'informations d'Edifixio repose sur de nombreuses applications internes, conçues sur l'écosystème de Salesforce. Cela permet d'éviter de payer des solutions spécialisées, tout en bénéficiant d'applications personnalisées répondant aux besoins spécifiques de l'entreprise.

Assigner des stagiaires et des alternants aux projets internes présente de multiples avantages, tant pour l'entreprise que pour les apprentis. Actuellement, nous constatons une demande limitée en matière d'emplois qualifiés sur Salesforce. Recruter des jeunes permet à la fois de leur offrir une opportunité de se former par la pratique, tout en disposant d'une main-d'œuvre active sur ces projets.

Nous verrons dans un premier temps l'environnement de travail professionnel, de la présentation complète de l'entreprise à l'explication pratique de son fonctionnement. Je décrirai dans un second temps le projet pour lequel je suis assigné, ainsi que son environnement technique et organisationnel. Dans un troisième temps, je présenterai en détail mes missions sur chacune des applications. Pour finir, je ferai un bilan sur professionnel et personnel tout en abordant les différentes difficultés que j'ai rencontrées.

I) Présentation de l'environnement professionnel

Vue d'ensemble sur l'entreprise, mon équipe et les différentes activités qui composent une journée de travail.

I. A) L'entreprise Edifixio

« Venez grandir avec nous »



Figure 1: Logo d'Edifixio

Edifixio est une Entreprise de Services du Numérique (ESN)¹ créée en 2000 et comptant 314 employés en France².

L'entreprise propose de nombreuses solutions Business to Business³, **en majorité tournées vers la donnée et les services en ligne** : gestion de relation client, stratégie numérique, intelligence artificielle, sans oublier l'intégration cloud.

La majorité des employés français travaillent au siège social situé dans le département des Hauts-de-Seine (voir carte dans l'Annexe A1 (page 39)). Edifixio possède aussi des bureaux à Grenoble, Nantes, Lyon et Rennes. Pour finir, la société possède également des antennes aux États-Unis (Boston), en Tunisie (Tunis) et en Inde (Calcutta et Bangalore) [1].

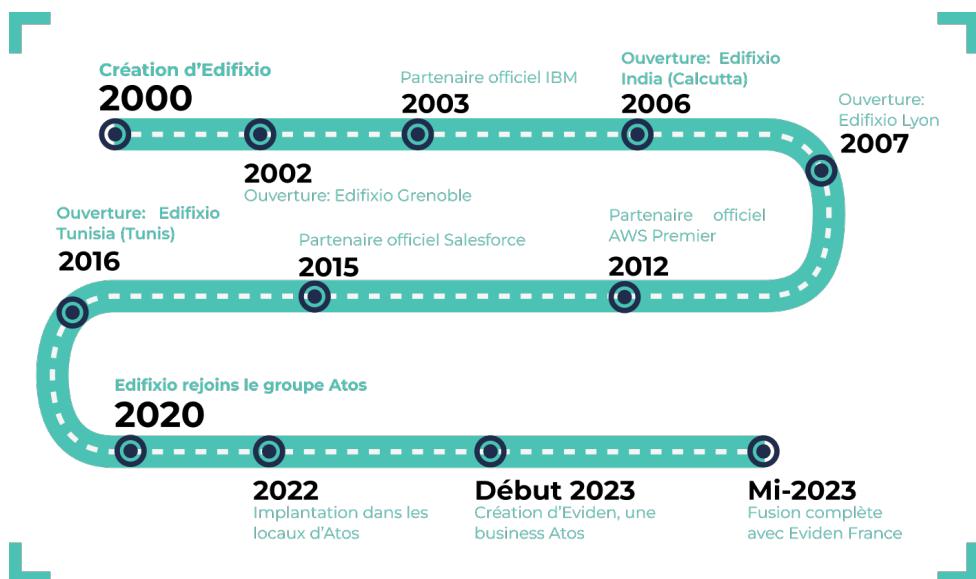


Figure 2: Dates clés d'Edifixio

I. B) Projet de fusion avec la société mère, « Eviden »

Fin 2020, Edifixio a été rachetée par la société Atos [2]. L'entreprise, auparavant implantée à Levallois, déménage plusieurs kilomètres au nord, à Bezons.

En 2022, Atos subit une baisse en bourse et pour compenser les pertes, l'entreprise envisage de se scinder en deux entités distinctes :

- « Atos TFCo », qui regroupera les activités traditionnelles telles que l'infogérance et les espaces de travail numérique.
- « Eviden », qui rassemblera les activités liées à la cybersécurité et au cloud.

C'est dans cette dernière entité que sera intégrée Edifixio durant l'été 2023.



Figure 3: Logo d'Atos et d'Eviden

¹Anciennement connu sous l'appellation SSII, une ESN vend des services numériques (audit, développement) à une autre entreprise, contrairement à un éditeur de logiciel qui crée puis vend son propre logiciel.

²Chiffres internes pour avril 2023 et dénombrant uniquement les employés travaillant en France.

³Anglicisme pour « commerce interentreprises » : Lorsque la clientèle d'une entreprise est exclusivement constituée de professionnels.

Ainsi, Edifixio est au cœur d'un lourd processus de conduite au changement depuis fin mars 2023. Les employés s'alignent progressivement sur les règles Atos (accords télétravail, règles de congés...) et adoptent les nouveaux outils. **Ayant travaillé cette année sur l'intranet d'Edifixio, la conduite au changement a fortement impacté le travail à réaliser ainsi que nos priorités.**

I. C) Partenariats : Expertises et clients

Edifixio possède multiples expertises grâce à ses partenariats avec de nombreux acteurs du cloud, tel que :

- **Salesforce CRM** : Une plateforme cloud⁴ SaaS⁵ spécialisée dans les logiciels de relation client. Titulaire d'un partenariat depuis 2020 [3], cette expertise figure parmi les raisons du rachat de notre entreprise par Atos.
- **IBM Cloud / Red Hat** : Une plateforme de préférence pour héberger l'infrastructure informatique de grandes entreprises, proposer des services d'IA ou traiter de grands volumes de données.
- **Amazon Web Services** : Fort de son vaste catalogue de services (stockage, données, cloud...), on estime qu'un tiers du marché cloud est détenu par AWS [4]. Edifixio possède une forte expertise dans le conseil et l'intégration des systèmes.
- **Microsoft Azure** : Une plateforme cloud très utilisée pour faire du traitement de données, héberger une infrastructure et des applications dans un serveur web.



Figure 4: Logos des 4 plateformes cloud maîtrisées

En plus de partenariats techniques, Edifixio possède un portefeuille de clients bien établis et venant de secteurs variés. Nous pouvons citer Renault, Schneider, LVMH, Fnac/Darty... Durant mon année d'alternance, je n'ai pas travaillé en clientèle, seulement sur des projets en interne.

I. D) Le pôle Digital Platform

Les employés d'Edifixio sont assignés à un pôle, chacun étant lié à une compétence technique ou un besoin. **J'ai intégré le pôle Digital Platform**, qui compte une centaine d'employés. Les services proposés sont essentiellement la création sur mesure d'outils marketing, ainsi que la maintenance et l'évolution de solutions avec le logiciel Salesforce.

Info

Je présente en détail le logiciel Salesforce dans une section dédiée de la partie II

Dans chaque équipe de projet, on y trouvera deux profils type :

- Les *business analysts* Salesforce, qui possèdent généralement la double casquette « administrateur » et « consultant fonctionnel » : ils sont capables de comprendre les besoins et les exigences du client pour configurer le logiciel et proposer des solutions.
- Les développeurs Salesforce, qui peuvent créer du code sur mesure lorsque le besoin ne peut pas être effectué avec l'interface graphique de base.

I. E) Organigramme

Les différents pôles sont appelés en interne des « practices », j'utiliserai régulièrement ce terme tout au long du rapport. Voici l'organigramme fonctionnel durant mon année scolaire :

⁴Cloud : Service proposant des serveurs informatiques à distance pour stocker, traiter et gérer des données. Cela évite à une équipe de devoir acheter et administrer son propre matériel.

⁵SaaS : pour Software as a Service — proposer un logiciel en ligne installé sur le cloud, accessible par un navigateur internet. Prêt à emploi, il n'y a pas besoin d'installer le logiciel sur un ordinateur.

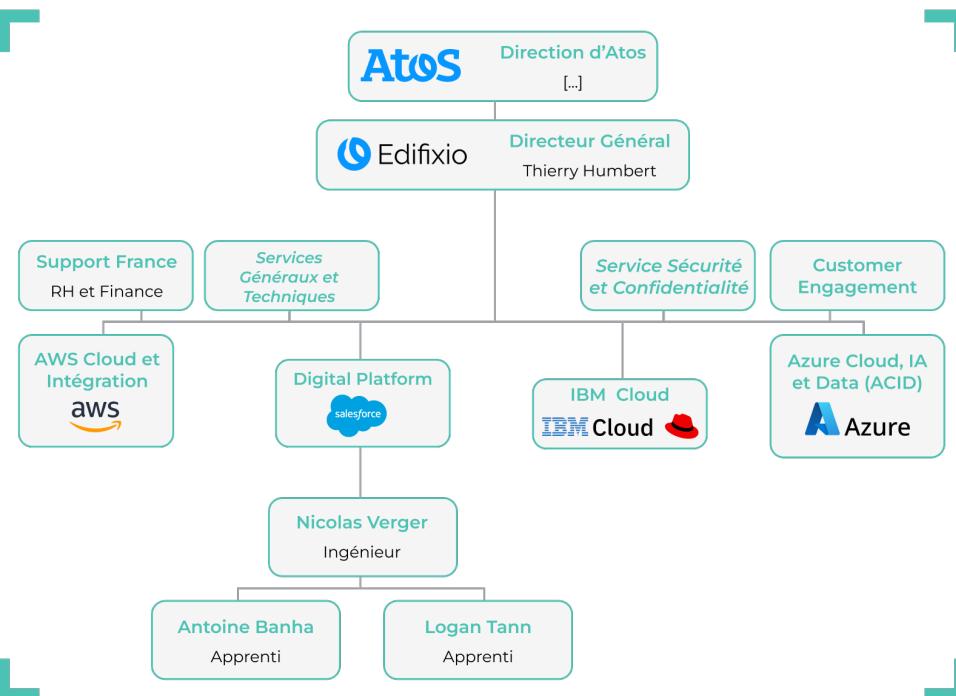


Figure 5: Organigramme fonctionnel

I. F) Vie en entreprise

○ Les locaux



Figure 6: Photographie à l'extérieur et à l'intérieur des bureaux Atos

Edifixio est implantée au siège social d'Atos, situé à Bezons (plusieurs kilomètres au nord du centre d'affaires de *La Défense*). Faire son premier jour dans une multinationale est particulièrement surprenant, le style des locaux est typique des bureaux commerciaux que l'on voit dans les séries.

À leur arrivée, tous les nouveaux employés reçoivent un matériel de fonctions neuf, ainsi que quelques goodies (une batterie portable, des stickers...)



Figure 7: Photographie de mon espace de travail

Nous travaillons dans un open-space. Chaque poste de travail étant doté de deux écrans, d'une connexion internet câblée et d'une chaise ergonomique. L'environnement est donc pensé pour être confortable et je n'ai jamais eu de douleurs physiques malgré toutes ces heures passées devant l'écran. L'étage est aussi doté de quelques salles de réunion ainsi que d'un espace avec des canapés.

○ Une journée type

La journée type d'un développeur chez Edifixio dépend surtout du projet sur lequel nous sommes assignés. Par exemple, certaines équipes démarrent toujours leurs journées par une réunion quotidienne de feedback (les « daily »). De notre côté, sans compter les divers événements de vie d'entreprise (que je détaillerai juste après cette section), **notre journée est essentiellement dédiée au développement de nos tâches.**

Les employés d'Edifixio ont un statut de cadre et bénéficient donc d'une grande autonomie. Nous sommes libres de choisir nos horaires de travail. Pour ma part, j'ai pris pour habitude de travailler en présentiel de 09:30 à 18:00, et prendre une pause repas à 12:00.

○ Les évènements RH

Depuis l'année dernière, deux événements RH sont réalisés chaque année :

- **Le calendrier de l'avent Edifixien** : Un calendrier de l'avent numérique où chaque jour, il est possible d'ouvrir une case pour découvrir une surprise ou un défi. Les employés sont invités à partager leur expérience sur le réseau social d'entreprise.
- **La chasse aux œufs Edifixienne** : Un format complètement revu cette année. Des QR codes sont cachés dans tout l'open-space. Avec une application, les employés doivent chercher les 16 codes et peuvent participer à un tirage au sort pour gagner des chocolats Pierre Hermé.

Info

Ces deux applications web ont été développées par moi-même et Antoine (Apprenti à l'EFREI en L3 LSI-1) en moins d'une semaine.

En raison de mes nombreuses missions, je n'ai pas souhaité détailler mon travail sur ces applications. Pour en savoir plus, se référer au rapport d'apprentissage d'Antoine Banha.

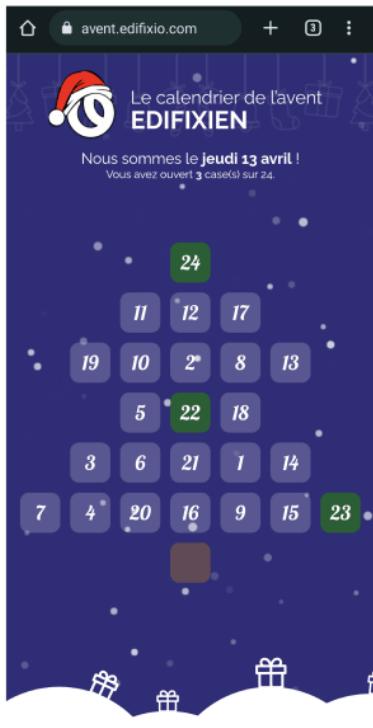


Figure 8: De gauche à droite : L'application « Calendrier de l'avent Edifixien » et « Chasse aux œufs Edifixienne »



Figure 9: Essai de l'application pâques sur un vrai QR code

○ Les sessions de Coding dojo

Un **Coding dojo** est une session mensuelle d'apprentissage en équipe (« workshop »), visant à résoudre collectivement un défi de programmation (typiquement, un problème algorithmique). À la fin de la séance, les équipes présentent leurs solutions, les difficultés rencontrées, ainsi que leur ressenti sur la session et les possibilités d'amélioration pour les prochaines séances. Le feedback est ensuite archivé de manière textuelle.

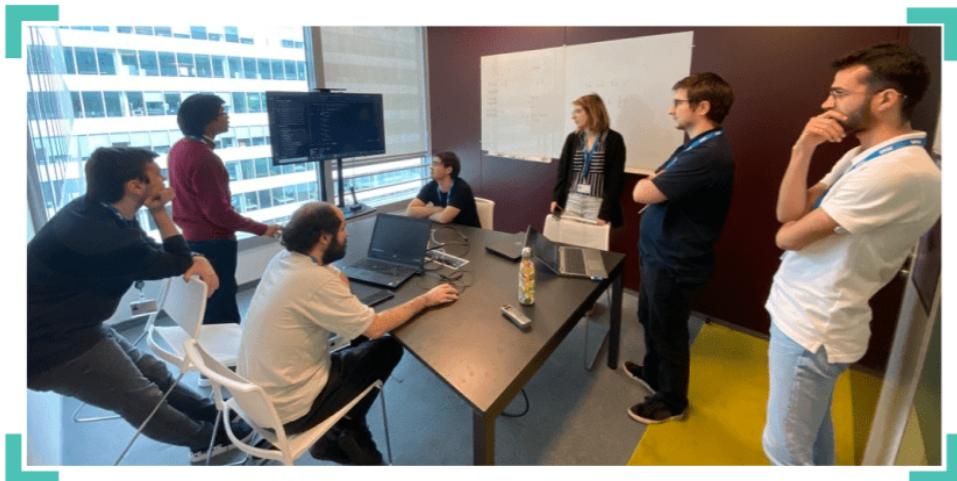


Figure 10: Une session de Coding Dojo

Le format habituellement pratiqué (« randori kata ») prône la découverte de méthodologies telles que le *Test Driven Development* (TDD⁶) et le *pair programming*⁷. Certains mois, nous pratiquons différentes variantes telles que :

- Le prepared kata : les présentateurs montrent la solution au tableau. Les participants doivent la reproduire étape par étape et poser des questions au besoin.
- Les quick katas : Exercices types présentés durant les entretiens d'embauche. Il est question de privilégier la pratique pure plutôt que l'apprentissage de méthodologies.
- Les refactoring katas : L'exercice a une solution qui fonctionne, mais qui peut être améliorée. Les participants doivent la perfectionner.

Info

Si les termes sonnent japonais, c'est parce qu'ils sont directement inspirés du Judo [5]. Un défi se résout en binômes tournants de façon régulière (“le randori”), au sein d'une salle de réunion qui ne laisse pas lieu à la distraction (“le dojo”). Le sujet (“kata”) peut être répété pour se perfectionner.

Un Coding dojo est une bonne occasion de découvrir de nouveaux langages de programmation. Il s'agit généralement de sujets de logique, donc la solution n'est pas limitée par une technologie particulière.

À première vue, organiser des dojos peut sembler être de la perte de temps. Mais, au fil des séances, il constitue un réel intérêt par sa capacité à former de manière ludique et efficace. Avec Antoine, nous sommes responsables de l'organisation et de l'animation de ces séances de team-building.

⁶TDD : méthode de développement consistant à rédiger un *Test Unitaire* (TU) avant de coder la fonctionnalité et sa logique. Si le concept de test unitaire n'est pas une notion acquise par le lecteur du rapport, se référer à l'annexe Anexe A5 (page 41)

⁷Pair Programming : codage en binôme, avec des rôles tournants

○ Les sessions de Knowledge sharing

Une session de *Knowledge sharing* est une réunion bimestrielle très similaire au concept de conférence ou webinaire. Une équipe d'employés présente un sujet en visioconférence, qu'il soit précis (ex : présentation technique des dernières fonctionnalités de Salesforce) ou plus ouvert (ex : table ronde sur certaines méthodologies).

La session se déroule habituellement en anglais et est écouteée par tous types de profils (développeurs comme business analysts). Il faut donc veiller à la rendre à la fois attractive et accessible à tous.

Organiser des sessions présente plusieurs avantages. Pour le présentateur, cela permet d'approfondir ses connaissances ou de les perfectionner, et pour les spectateurs, c'est une occasion d'apprendre des expériences des autres.

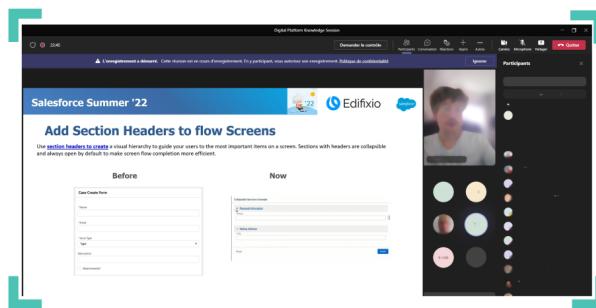


Figure 11: Une session de Knowledge Sharing

○ Sorties d'entreprise

Pour finir, des *afterworks* ou soirées barbecue organisées entre collègues. Plus rarement, des événements sont parfois organisés par l'entreprise et le CSE.

Par exemple, une visite et un buffet avait été organisée au musée Rodin (qui a été client d'Edifixio). Plus récemment, un atelier fictif de cadrage de projet client⁸ dans une salle de travail réservée, suivi d'un *action-game* au Koezio de Lieusaint.



Figure 12: Photographie de l'équipe Salesforce au musée Rodin



Figure 13: Atelier interactif de cadrage de projet avec l'équipe AWS

⁸Cadrage de projet : Dans le contexte de l'exercice, identifier les forces, faiblesses, risques et opportunités d'une mise en situation tant du côté « chef de projet » que du côté « client ». Un véritable cadrage de projet inclus l'identification des ressources financières, les contraintes de temps, les objectifs et critères chiffrés de réussite...

II) Écosystème technique

Présentation du contexte et de l'environnement technique pour lequel j'ai effectué mes missions

II. A) L'écosystème Salesforce

L'objectif de mon parcours d'apprentissage chez Edifixio est de me préparer à devenir un ingénieur logiciel avec **spécialisation sur la plateforme Salesforce**.

○ Qu'est-ce qu'un logiciel CRM (Customer Relationship Management) ?

Un CRM est l'acronyme anglais de *logiciel de gestion de relation client*. Il permet de centraliser et d'organiser les données des clients (contacts, historiques d'achats, interactions) plutôt que de les inscrire dans un fichier Excel ou un répertoire. Cela permet de générer des rapports, fournir des outils de collaboration ou de support client, ou bien de l'automatisation pour simplifier des tâches répétitives.

Mais de la même manière qu'une maison ne se limite pas qu'à sa cuisine, Salesforce propose bien plus d'outils que proposerait un simple CRM (tel qu'un réseau social d'entreprise ou des outils de marketing intelligents [6]).

○ Salesforce

Né en 1999, Salesforce est un logiciel commercial sur le cloud permettant de gérer la relation clientèle d'une entreprise. Il est considéré comme un pionnier du modèle dit « Software as a Service » (SaaS) [8]. Contrairement aux logiciels classiques, il n'est pas nécessaire de l'installer localement. Les utilisateurs peuvent simplement se connecter via leur navigateur pour travailler depuis n'importe où.



Figure 14: Différentes fonctionnalités d'un CRM

Source illustration : [7]

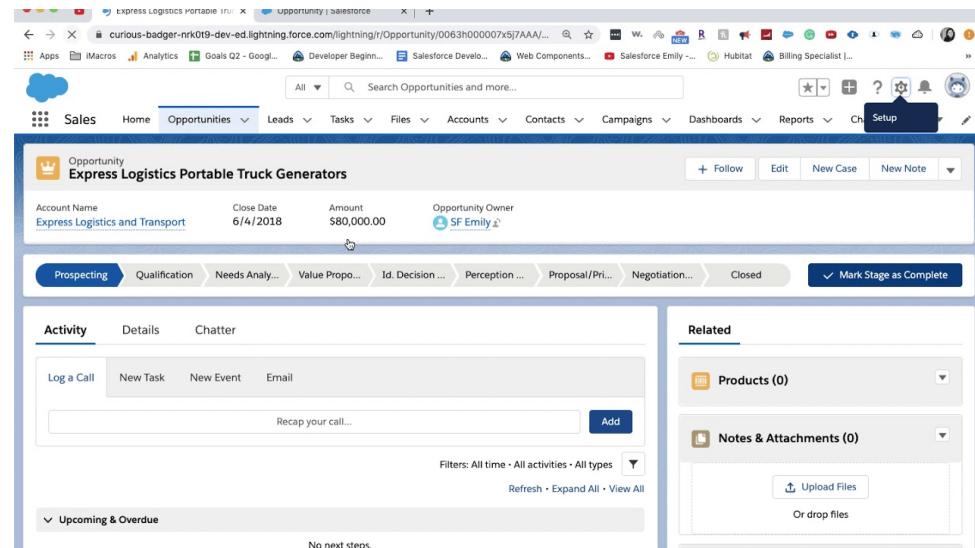


Figure 15: Capture d'écran de l'interface Salesforce

Avant de rejoindre Edifixio, j'avais déjà créé plusieurs applications full-stack en utilisant PHP et Node.js, mais je ne savais pas en quoi consistait l'écosystème Salesforce. Ce n'est pas surprenant, car Salesforce est avant tout un logiciel CRM et ne se pratique que dans une finalité commerciale. Ce n'est donc pas une technologie à portée de tous.

En plus des fonctionnalités de base, il est possible d'intégrer nos propres programmes au sein même du logiciel. Cela représente un sérieux avantage comparé aux logiciels traditionnels, puisque l'entreprise peut faire appel à des développeurs spécialisés si les outils proposés par défaut ne permettent pas de couvrir des besoins spécifiques.

II. B) Le *Salesforce Interne*, système d'informations d'Edifixio

Edifixio possède une instance Salesforce pour stocker les données de ses clients ou pour suivre les candidatures pour les offres d'emploi. En plus des fonctionnalités de gestion clientèle classiques, l'entreprise a développé de nombreuses applications internes pour **baser l'ensemble de son système d'information sur cette plateforme**. Nous pouvons citer :

- Achats : L'outil de comptabilité, utilisé par la team finance.
- TRH : L'outil pour poser ses jours de congés.
- Tock : L'outil pour pointer le temps passé sur chaque projet.
- Core : Ce n'est pas une application en tant que telle, mais plutôt un package sous lequel sont stockées certaines parties de code conçues pour un usage général au sein de plusieurs applications.

Ces applications sont disponibles en français et en anglais. Pour en savoir plus, se référer à l'[Annexe A4](#) (page 40).

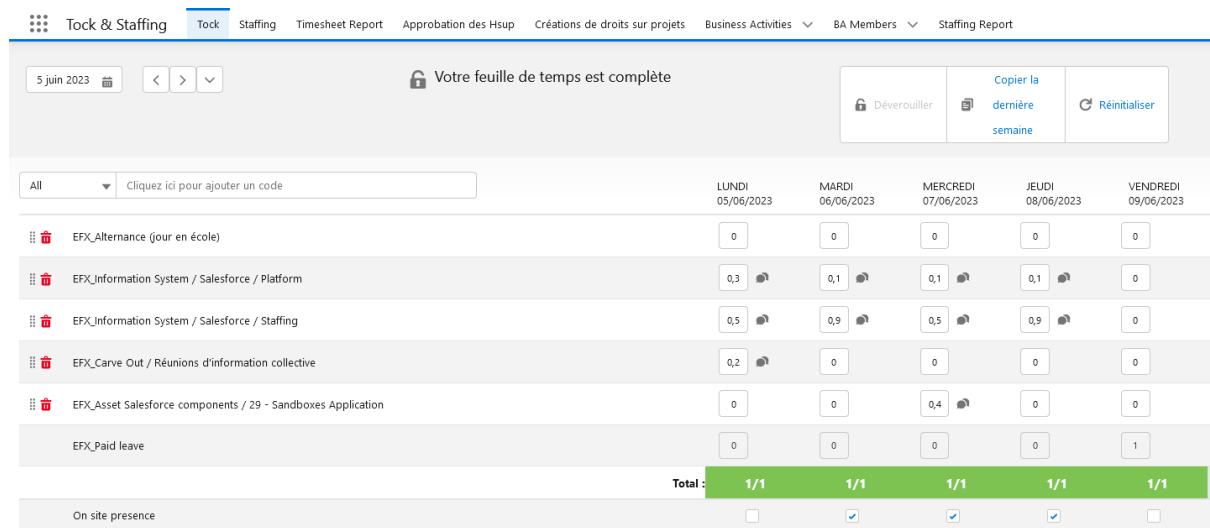


Figure 16: Aperçu de l'application « Tock », où je déclare la répartition de mon temps travaillé durant la semaine en cours.

En tant qu'apprenti-ingénieur, mes missions sont la création de nouvelles fonctionnalités et la maintenance évolutive (correction de bugs, améliorations...). Je détaille mes principales missions pour chacune des applications dans la [partie III du rapport](#).

L'entreprise développe également d'autres projets internes dont je n'assure pas la maintenance. Par exemple, le projet « ChangEs », qui permet de faciliter le déploiement de nos nouvelles fonctionnalités en production⁹. Il existe aussi le projet « SandboxEs », qui permet de transférer des échantillons de données ainsi que leurs dépendances entre deux sandboxes¹⁰.

Info

Le logiciel « SandboxEs » est en cours de développement et n'est pas encore utilisé. Je prévois d'y passer la majorité de mon temps de travail dès l'année prochaine, en tant que sujet de mémoire.

⁹Déploiement en production (parfois abrégé « mise en prod. ») : Action consistant à mettre à jour un logiciel en envoyant du code sur le serveur de production (accessible par les utilisateurs finaux, en opposition aux serveurs dits « sandbox » où les développeurs peuvent créer leurs fonctionnalités de manière isolée).

¹⁰Sandbox : serveur de développement complètement isolé et accessible uniquement par les développeurs du projet. Ceux-ci peuvent envoyer du code et faire des tests sans que cela impacte les utilisateurs finaux.

II. C) Organisation du travail à réaliser

○ Gestión del proyecto, documentación interna y almacenamiento del código

Depuis que l'équipe du Salesforce interne a complètement revu les méthodes de développement, les outils Jira et Confluence sont utilisés respectivement pour la gestion des tâches et la documentation interne. À chaque fois qu'un composant commun est créé, il faut le documenter dans le wiki Confluence.

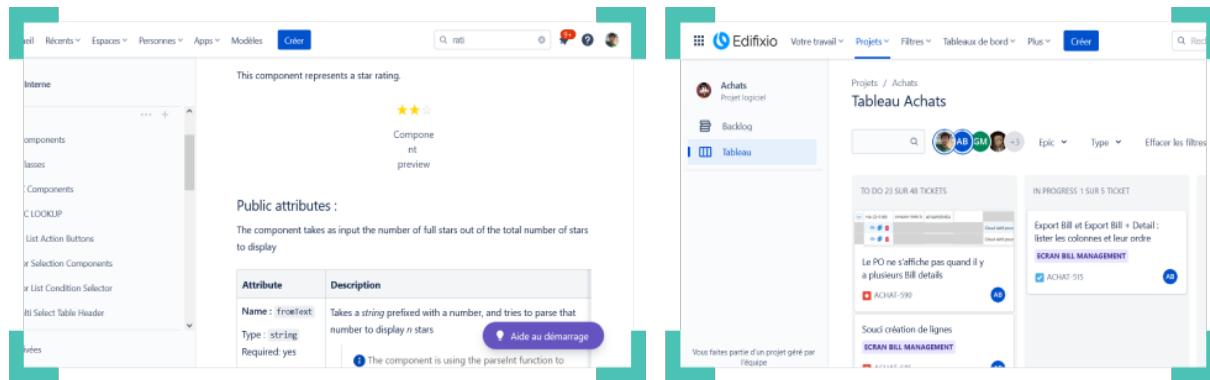


Figure 17: Captures d'écran respectives de la documentation interne sur Confluence et du tableau Kanban sur Jira

Pour stocker notre code sur le cloud et gérer les opérations de fusion de code, nous sommes dotés d'une instance GitLab Community Edition, hébergée sur les serveurs d'Edifixio. Il n'y a pas autant de fonctionnalités que GitLab Entreprise Edition, mais il remplit tout à fait ses fonctions et offre un bon niveau de contrôle.

Info

Toute communication écrite doit être rédigée en anglais, qu'il s'agisse de notre code source, des descriptions sur GitLab ou de la documentation sur Confluence. C'est une bonne pratique dans le développement, car des personnes ne parlant pas français peuvent être amenées à travailler sur notre code.

Pour finir, nous utilisons Microsoft Teams pour des communications plus directes, et un mail Outlook quand il s'agit de questions administratives ou être notifié sur des processus automatiques (tel qu'une erreur de mise en production)

○ Cycle de vie des tâches du projet

Les applications internes sont maintenues par de petites mises à jour à cycles réguliers. Pour cela, le chef de projet assigne au développeur de petites tâches, qui suivent un cycle de vie bien défini.

Il est facile de penser que le métier de développeur se limite à programmer la tâche, et l'envoyer directement en production. En réalité, plusieurs processus de qualité se glissent entre la finalisation de la tâche et la mise en production.

Pour organiser l'avancement d'une tâche, nous les ordonnancions sur Jira, un tableau kanban virtuel. Voici une infographie récapitulant leur cycle de vie :

Info

Les principes de versionnage de code (branches, merge, GitLab...) et de tableau Kanban étant supposées connues par les lecteurs de ce rapport, la section expliquant ces notions a été déplacée dans l'Annexe A7 (page 43).

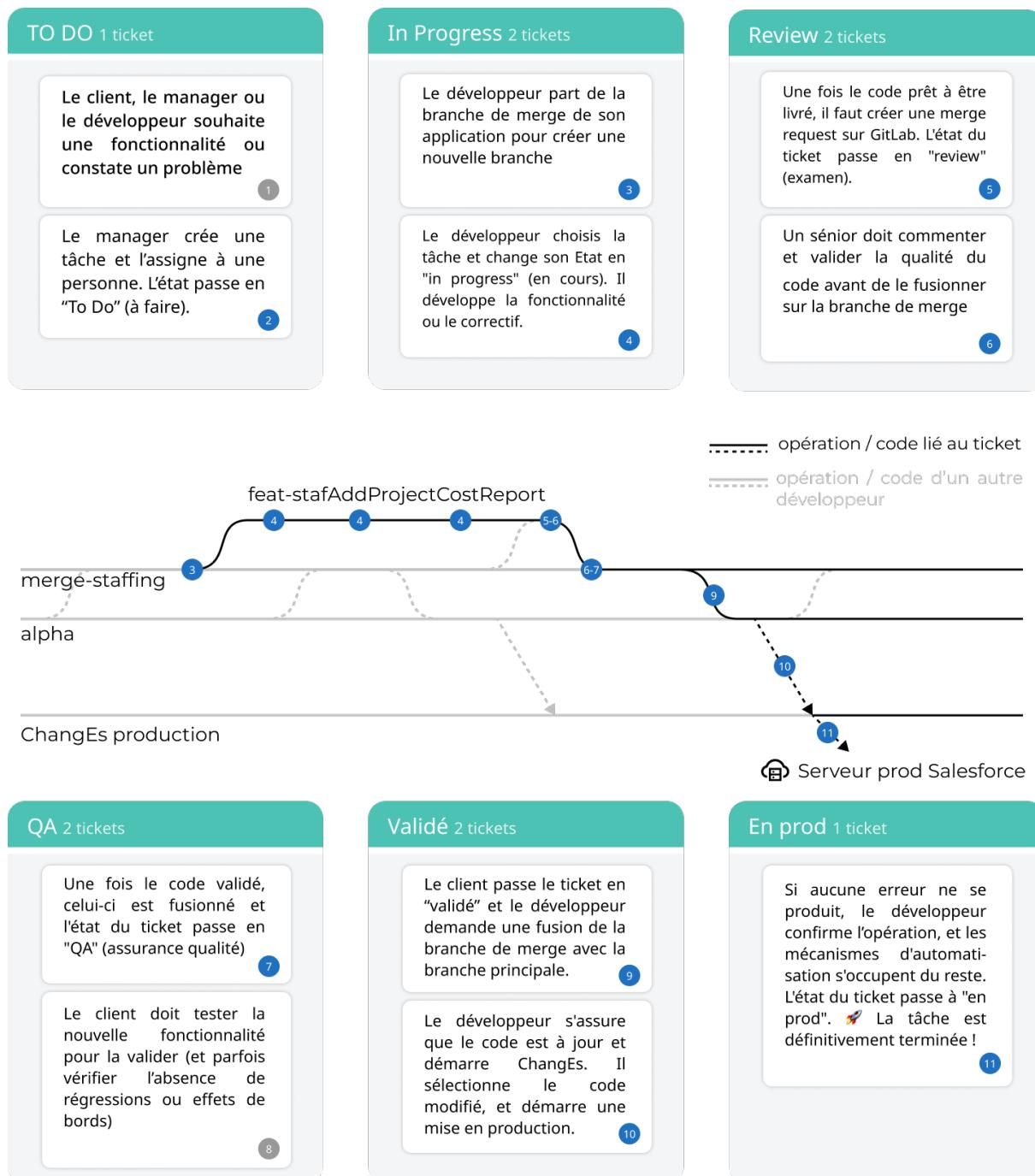


Figure 18: Cycle de vie d'un ticket et branches GitLab

II. D) Comment développe-t-on sur Salesforce ?

○ Les applications Salesforce (partie visuelle)

Salesforce est un CRM en ligne, si vaste que je ne comprends pas bien tous les produits et services proposés. Le cœur du logiciel est une base de données propriétaire, munie d'une interface sur lequel différentes applications peuvent se greffer.

Nous avons les applications Salesforce « standard », telles que *Marketing* ou *Customer Services*. Dans la plupart des cas, elles suffisent pour des utilisations basiques de CRM. Les administrateurs peuvent également installer des applications dans un marché nommé *App Exchange*. Enfin, il est possible de créer nos propres applications grâce à l'écosystème *Lightning Platform*.

Une application Salesforce est une collection d'onglets (que j'appelle « écrans », ou simplement « applications » par abus de langage). Nous pouvons créer différents types d'écrans :

- Les pages Lightning App. Avec une interface visuelle, les administrateurs peuvent créer des écrans en cliquant-déposant des widgets, sans avoir besoin de code. Nous ne l'utilisons pas sur l'intranet.
- Les pages objet. Elles affichent un enregistrement de la base de données, et l'administrateur peut ajouter divers boutons et widgets. Elles ressemblent à la Figure 15.
- Les pages de Composants Web Lightning. Un simple composant web est affiché en tant que page. Ce composant est créé avec le **framework Salesforce LWC**, en utilisant du code Javascript. Un exemple de code LWC commenté est disponible en [Annexe A3](#) (page 40).

Dans tous les cas, la création d'applications Salesforce et l'organisation des onglets se fera majoritairement sur l'interface graphique.

○ Programmation des automatismes (partie serveur)

Une base de données orientée objet est un type de base de données similaire aux bases de données relationnelles, comme MySQL ou Oracle. Les tables sont remplacées par des « objets » et les jointures par des « relations ». Elles ont l'avantage d'avoir une bonne expérience de développement, au prix d'une certaine lenteur (compter 60 secondes pour une requête avec des agrégations complexes).

Sur Salesforce, **on effectue nos requêtes avec le langage SOQL** (Salesforce Object Query Language), qui possède une syntaxe très semblable à SQL. Par exemple, pour afficher le nom du fournisseur de chaque produit, on fera la requête suivante : `SELECT monProduit.Fournisseur__r.Name FROM Produits__c`.

Apex est le langage de programmation back-end de Salesforce. Syntaxiquement, Apex est assez ressemblant à Java, mais son fonctionnement ressemble à des procédures stockées. Par exemple, il est possible d'exécuter des requêtes SOQL (dont le résultat sera retourné sous forme de liste) et des instructions dites « Data Manipulation Language » (DML, exemple : `insert monInstanceDobjet`; ajoutera l'objet dans la base de données) au milieu de notre code.

De telles possibilités seraient considérées comme une mauvaise pratique sur des langages traditionnels. Ce n'est pas vraiment le cas de Salesforce, où traitement des données est au cœur du langage. Tout de même, nous verrons comment adopter une architecture logicielle robuste pour corriger ce comportement.

```

1 public with sharing class AccountProcessor {
2     /**
3      * The method counts the number of Contact records associated to
4      * each Account ID passed to the method and updates the 'Number_of_Contacts__c'
5      * field with this value
6     */
7     @future
8     public static void countContacts(List<Id> listOfIds) {
9         List<Account> toProcess = [
10             SELECT Id, Name, Number_of_Contacts__c FROM Account
11             WHERE Id IN :listOfIds
12         ];
13
14         for (Account account : toProcess) {
15             account.Number_of_Contacts__c = [
16                 SELECT count() FROM Contact
17                 WHERE AccountId = :account.Id
18             ];
19         }
20         upsert toProcess;
21     }
22 }

```

1 Une requête SOQL de sélection de données.
2 Le résultat se manipule comme s'il s'agissait d'une structure C
3 Une instruction DML, pour pousser les changements vers la base de données. On le fait habituellement à l'extérieur d'une boucle pour des raisons d'optimisation

Figure 19: Un exemple de code Apex avec une requête SOQL et une instruction DML

On peut programmer très facilement un bouton ou un composant web pour qu'il appelle une méthode Apex. Il suffit que celle-ci possède la bonne annotation (`@AuraEnabled` pour être accessible depuis les composants web lightning, ou `@InvocableMethod` pour les boutons dans une page d'objet).

Pour finir, Apex nous impose d'avoir une couverture en tests unitaires d'au moins 70% avant d'envoyer son code en production. Pour en savoir plus, se référer à l'[Annexe A5](#) (page 41).

Info

Dans pas mal de projets clients, le code est remplacé par des « Flows Salesforce ». Ils utilisent une approche visuelle basée sur des diagrammes pour créer des processus sans nécessiter de code.

Ils sont donc compréhensibles par des profils « non devs », mais sont peu maintenables au fur et à mesure que les besoins grandissent. C'est pourquoi **nous ne les utilisons pas dans notre projet**.

Il existe également les composants Aura et les pages VisualForce, qui sont des ancêtres du framework LWC. Ces langages sont programmés sous la forme d'un fichier XML et encore utilisés sur une certaine proportion de projets clients.

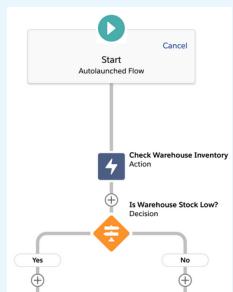


Figure 20: Un exemple de flow Salesforce

○ Comment intégrer de nouvelles fonctionnalités ?

Comme nous l'avons vu tout à l'heure, nous créons le code back-end avec le langage de programmation Apex, et l'interface graphique avec le framework Lightning Web Components.

Salesforce étant une plateforme cloud centralisée et gérant beaucoup de traitements à notre place, **le développement en local¹¹** n'est pas possible. Pour tester le code que nous avons écrit, nous sommes donc obligés de le téléverser et le faire compiler sur un serveur distant appelé « sandbox ».

Puisque chaque sauvegarde d'un fichier implique un déploiement sur ces serveurs, il faut donc attendre de 2 à 30 secondes pour voir effectivement nos changements, sans compter l'actualisation de la page sur le navigateur. Le temps d'attente impacte notre productivité, mais on s'y habitue rapidement.

Pour écrire et téléverser notre code, nous utilisons **l'IDE¹² IntelliJ Idea Ultimate** et l'extension tierce Illuminated Cloud. C'est la pile de logiciels privilégiée par l'équipe interne, car elle offre une expérience de développement supérieure. Ce n'est pas la seule manière, d'autres équipes utilisent un simple éditeur de texte, avec l'outil en ligne de commandes Salesforce DX pour téléverser le code.

Une fois que l'implémentation est fonctionnelle et validée par le client, le code produit peut être mis en production, livré à l'utilisateur final. Salesforce propose des outils de migration pour faire passer un code d'une instance de test à une instance de production.

En revanche, celle-ci est très manuelle et les risques d'erreur humaine sont grandes. Ainsi, les développeurs sont dotés d'une application interne nommée ChangEs, qui fait exactement ce travail.

Créée avec Angular et Javascript, cet outil de développement nous aide à récupérer facilement le code à mettre en production. Il assure également un historique du code déployé et gère tout un déroulement d'opérations à notre place.

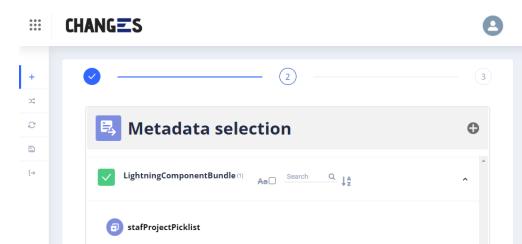


Figure 21: Interface de ChangEs v2, pour sélectionner les classes à déployer en production

¹¹Développement en local : Être capable de stocker et démarrer le code d'un projet depuis son propre ordinateur.

¹²IDE (Integrated Development Environment) : Un éditeur de texte doté d'outils supplémentaires pré-installés pour l'analyse complète (indexation), le débogage et le déploiement du code. Plus puissant, mais consomme aussi beaucoup plus de ressources dans l'ordinateur.

II. E) Architecture logicielle

○ Structure de fichiers d'un projet sur Salesforce

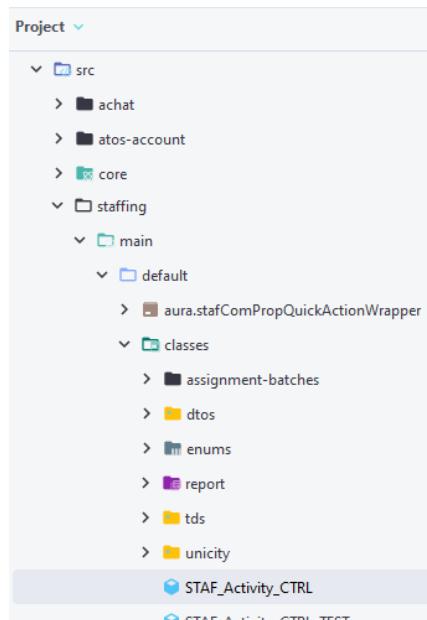


Figure 22: L'arborescence du projet, avec une classe de l'application Staffing sélectionnée

La structure des fichiers respecte les conventions suggérées par Salesforce. Dans le répertoire git du Salesforce interne, chaque application est représentée par un package : achat, core, skills, staffing, tock, etc.

Un package est composé de plusieurs dossiers, chacun stockant un type de métadonnées bien précis. **Nous pouvons citer le dossier lwc qui contient tous les composants Web Lightning, ou le dossier classes, qui possède le code Apex** nécessaire pour faire fonctionner le back-end. Tous ces dossiers ont leur propre architecture, puisqu'un langage de programmation est attribué à chacun.

Le package core est assez spécial, car il ne représente pas une application. Il contient des composants et du code généraliste, utilisé au sein de plusieurs applications. Par exemple, une fonction pour générer un fichier CSV, ou bien un composant permettant de sélectionner une liste d'utilisateurs.



○ Architecture d'un composant front-end sur LWC

LWC est un framework qui permet de créer des composants web. Autrement dit, des briques indépendantes qui vont former notre application (ex : fenêtre contextuelle, tableau de données, page complète composée elle-même d'autres composants).

L'architecture de LWC est assez simpliste : **un composant est représenté par un dossier, avec un fichier HTML** (squelette), **CSS** (design) et **Javascript** (logique). En revanche, par souci de performance et de choix techniques, LWC impose beaucoup de restrictions au niveau de la syntaxe et de l'architecture.

Il n'y a pas beaucoup de concepts d'architecture logicielle à appliquer, il faut surtout veiller à séparer le code HTML et déléguer les responsabilités aux composants enfants. Chaque élément ne devrait pas dépasser les 300 lignes de code. J'ai déjà rencontré des fichiers contenant plus de 2500 lignes, et cela rend le travail extrêmement difficile. C'est pourquoi il est primordial d'accorder une grande importance à la qualité du code qui est produit.

○ Classes apex et architecture logicielle en couches (Clean Architecture, Domain)

Le langage Apex a été pensé pour remplacer les concepts de procédures et triggers¹³ que nous trouvons sur les bases de données classiques. C'est pourquoi le modèle promu par Salesforce est de tout développer sur une même méthode.



Par exemple, si nous avons besoin de récupérer les personnes qu'un employé manage, il suffit de créer une fonction allant faire une requête SOQL sur la base de données, puis retourner la liste sans traî-

¹³Procédure stockée et triggers : Ensemble d'instructions pré-compilées qui sont stockées dans le système de gestion de base de données. Les procédures stockées sont exécutées sur demande, tandis que les triggers sont exécutés automatiquement lorsqu'une donnée est modifiée.

tement supplémentaire. Cependant, dans un outil aussi riche que le Salesforce Interne, les besoins métiers sont complexes et soumis à de nombreuses évolutions.

Il faut donc faire appel à nos connaissances d'architecture logicielle et trouver les modèles adaptés à notre besoin. Un ancien collègue et étudiant de l'EFREI (Arsène Lapostolet) s'est penché sur la question et a dédié cette problématique dans le cadre de son mémoire de fin d'études, « Clean Architecture applied to Salesforce ».

La solution, dans les grandes lignes, est d'avoir trois couches indépendantes :

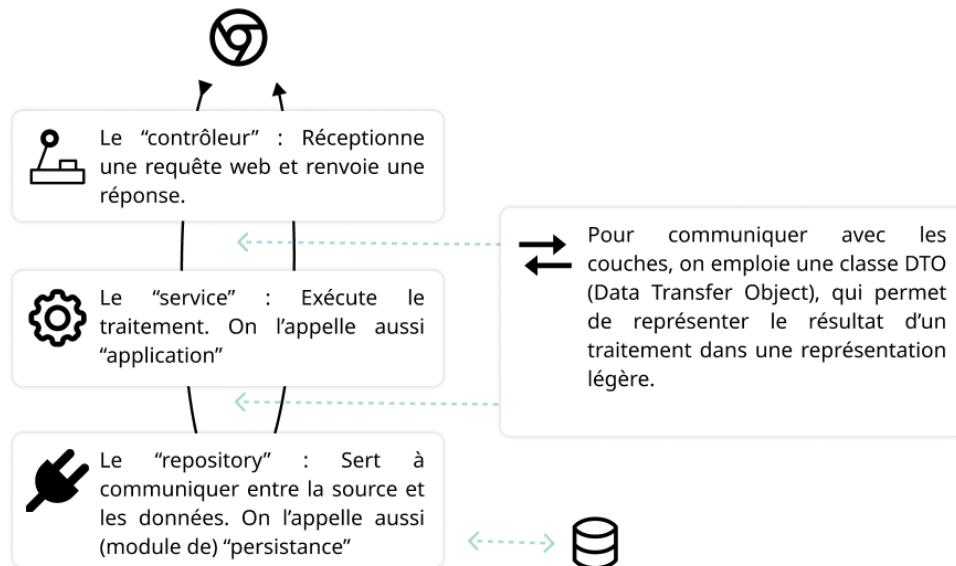


Figure 23: Domain pattern en trois couches (contrôleur, service et repository)

En interne, nous pouvons tirer parti de cette architecture pour pouvoir réutiliser la logique (service) sans dépendre d'un contrôleur. Ou bien remplacer un module d'accès aux données (« repository ») par un système de cache. Ce modèle a été très utile sur l'application « Staffing Report », je le détaillerai donc comment l'utiliser en pratique sur [la section dédiée \(III.D > Staffing Report\)](#).

Info

Dans son mémoire, Arsène désigne ce modèle comme une application des principes de Clean Architecture, sans spécifiquement donner de nom.

Par abus de langage, **nous appelons cette méthode le « Domain Pattern »**, car c'est le modèle choisi lorsqu'une équipe souhaite appliquer les principes du « Domain Driven Design »¹⁴.

J'appelle souvent cette structuration « Architecture logicielle multicouche », ou « 3-tier architecture ». Je trouve que cette dénomination est plus adaptée, mais elle a aussi le défaut d'être plus généraliste.

En effet, certains articles [9] peuvent considérer les trois couches comme l'affichage navigateur, le traitement serveur et la disposition d'une base de données, alors qu'il ne s'agit pas de notre modèle.

Ce modèle n'est pas non plus à confondre avec le MVC (Model-View-Contrôleur)¹⁵.

¹⁴DDD ou « Conception pilotée par le domaine » : une approche de conception logicielle qui met l'accent sur une modélisation correcte des besoins métiers et qui privilégie la logique du domaine appliquée plutôt que de se focaliser sur des choix technologiques.

¹⁵Pour en savoir plus, voir la [section dédiée sur la page wikipedia Modèle-Vue-Contrôleur](#)

III) Détail des missions

Présentation des missions effectuées durant l'année scolaire et mise en pratique des concepts théoriques au travers des différentes applications composant mon projet

III. A) Mon travail en tant qu'apprenti-ingénieur

Ma mission d'alternance consiste essentiellement à assurer la maintenance évolutive de l'intranet d'Edifixio, qui repose sur l'écosystème propriétaire Salesforce. Cela peut être **l'implémentation de nouvelles fonctionnalités, la correction de bogues ou la réécriture de fonctionnalités obsolètes**. La gestion de projet est basée sur le principe Kanban : chaque employé est affecté à quelques tâches courtes et indépendantes (des "tickets") ayant un cycle de vie (To-do > In progress > Review > QA > Validé > En prod)

Les fonctionnalités sur lesquelles j'ai été assigné consistent généralement à stocker de manière simplifiée des données différentes mais possédant de fortes relations, ainsi que de générer de rapports depuis ce lot de données. Par exemple : développer un code permettant de calculer la rentabilité d'un projet, en récupérant les employés, leur fiche de temps et le salaire qui leur sont associés.

Contrairement aux croyances, **un développeur full-stack** n'est pas seulement une personne capable de créer à la fois du code côté front-end (LWC) et back-end (Apex). Il **doit également être capable de gérer toutes les étapes de la réalisation d'une tâche**, de la couverture de tests au déploiement en production.

Les langages de programmation maîtrisés ne sont qu'un bagage d'outils pour pouvoir assurer le cycle de vie d'une mission :

- **Faire des choix de conception logicielle** adaptée au besoin de la mission.
- **Implémenter** la fonctionnalité **avec un code de qualité** (maintenable, facile à faire évoluer et facile à lire). Cela ne constitue que 50% de notre temps.
- Couvrir la fonctionnalité en **tests automatisés** (cf. [Annexe A5](#) (page 41)). Une tâche chronophage, mais nécessaire.
- Si besoin, l'application doit supporter le français et l'anglais (cf. [Annexe A4](#) (page 40))
- Passer la **revue de qualité** de code (exemple : [Annexe A2](#) (page 39)) et revue fonctionnelle
- Produire la **documentation technique** (exemple : [Annexe A6](#) (page 42))
- **Déployer** le code en production.

En plus de l'activité principale sur l'intranet d'Edifixio, nous avons pris part à de nombreux projets transverses, tel que l'organisation d'ateliers de team building et le développement d'applications dédiées aux évènements annuels.

III. B) L'outil de comptabilité (Achats)

« Achats » est l'outil de comptabilité interne d'Edifixio, qui propose deux sous-applications (« écrans ») riches en fonctionnalités.

Bill management est l'application qui permet de gérer l'ensemble des factures internes d'Edifixio, tel que l'achat de matériel, de services cloud ou des chocolats pour pâques...

Sur un simple écran ressemblant à un tableau Excel, l'équipe finance peut enregistrer les achats grâce à des champs bénéficiant d'autocomplétion. L'outil supporte différentes devises monétaires, la gestion des taxes, des sous-factures, des dates d'échéances, des pièces jointes, des exports, etc.

Réception est l'application permettant d'enregistrer la réception de produits et organiser l'étalement des paiements sur le temps. J'ai eu l'occasion d'améliorer l'interface graphique et d'ajouter quelques fonctionnalités d'export.

| Achats | | | | | | | | | | |
|---------------------|------------------|-------------|--------------|--------------------------|------------------|--------|------|--|--|--|
| Procurement | | | | | | | | | | |
| | Bill Management | POs | PO Details | Achat Reception | Lettage Creation | Bills | More | | | |
| Σ 281154 | 2 | 2023 | C 1 2 3 > 50 | 16/05/2023 15/05/2023 | 30,00 € 6,00 | 105,37 | VAT | | | |
| HA-23-0184 -Z2-0132 | Tickets resto... | Po-01-01-01 | Supplier Ref | Accounting Date EDI Date | Amount | Excl | | | | |
| HA-23-0180 -Z2-0034 | Tent 123 | 25/05/2023 | 24/05/2023 | 526,37 € | 111,82 | 22,36 | | | | |
| | Cloud AWS p... | | | | 416,55 | 82,91 | | | | |
| HA-23-0186 -Z2-0132 | 888888 | 11/06/2023 | 26/05/2023 | 220,00 € | 0,00 | 0,00 | | | | |
| HA-23-0187 -Z2-0225 | AA | 11/06/2023 | 01/06/2023 | 22738,82 € | 0,00 | 0,00 | | | | |
| HA-23-0188 -Z2-0121 | Support CTRL... | 11/06/2023 | 01/06/2023 | 15000,00 € | 3000,00 | 0,00 | | | | |
| HA-23-0189 -Z2-0114 | 9999999999 | 11/06/2023 | 01/06/2023 | 49108,16 € | 0,00 | 0,00 | | | | |

Figure 24: L'application « Bill management »

| | |
|-----------------------------|-----------------------|
| Client | Team Finance |
| Sous-applis | 2 applications, 1 API |
| Développeurs | 2 à 3 personnes |
| Périodes travaillées | janvier, mars 2023 |

○ Etude de cas

L'accélération de notre fusion avec Atos a provoqué chez la team finance un changement des processus comptables. Dès décembre 2022, la priorité est donnée à l'export de lots de facture sur le logiciel utilisé par Atos (SAP accounting).

Le hic : leur logiciel ne possède pas d'API¹⁶, toute importation doit être effectuée à la main avec un fichier Excel respectant un format particulier. Pour combler le tout, Salesforce n'est pas capable de générer des fichiers Excel (.xlsx) nativement.

○ Solution

La solution choisie a été de créer un microservice avec un langage de programmation différent, dont la seule utilité est de récupérer des factures en entrée, et donner un fichier Excel en sortie.

Hébergée sur AWS, cette application a été écrite avec le langage de programmation C# et la librairie Apache POI. Au clic sur le bouton d'export, un service Apex récupérera les données, les traitera pour avoir un format adapté, puis enverra les données via une requête web à notre application externe. Il suffira de renvoyer à l'utilisateur le résultat de cette requête pour qu'il télécharge les fichiers Excel.

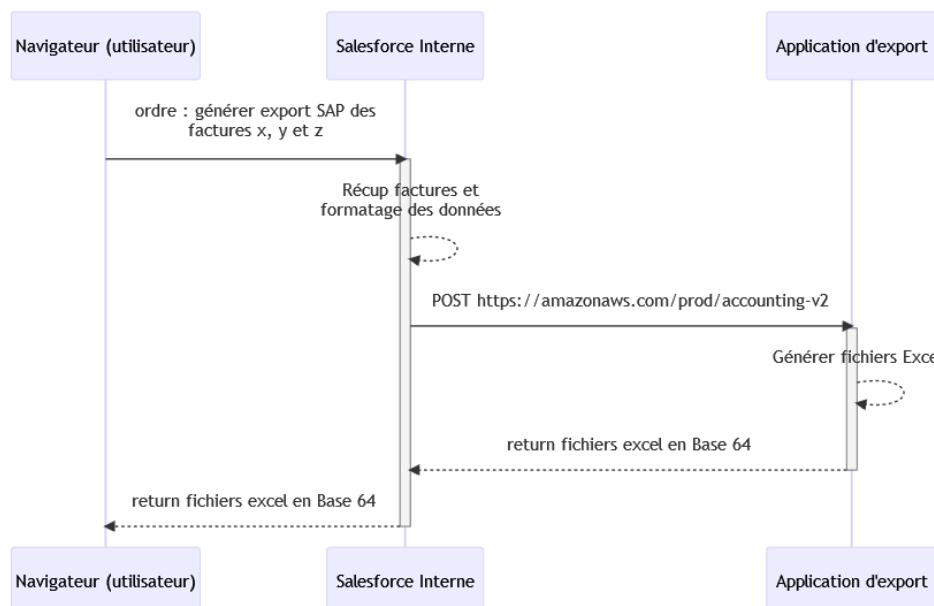


Figure 25: Diagramme de séquence simplifié représentant la solution choisie

○ Résultats

Créer une application dédiée rien que pour générer des exports est souvent une solution de dernier recours, car coûteux en temps et en ressources. Contrairement à Atos, l'équipe d'Edifixio est dans l'optique d'optimiser la productivité des employés, plutôt que devoir faire traiter ou importer les documents de manière humaine (tâches qui sont par ailleurs souvent délocalisées en Inde et en Bulgarie.)

Durant mon alternance, les besoins ont été régulièrement revus, tel que la gestion des devises. Malgré quelques difficultés de lancement, j'ai pu faire évoluer dans l'application dans les délais demandés. Elle est d'ailleurs sollicitée par d'autres outils, tel que IPM, utilisé par les chefs de projets pour saisir les productions mensuelles de chaque projet.

¹⁶ API (Application Programming Interface) : interface / service web qui permet à une application de pouvoir se connecter à une autre application. Dans notre exemple, nous aurions besoin d'une façade sur le logiciel comptable d'Atos qui permettrait à notre application Salesforce de pouvoir y importer nos données sans intervention humaine.

III. C) Les outils de ressources humaines (*Tock* et *TRH*)

TRH et *Tock* sont des outils de gestion de ressources humaines, et sans doute les deux applications les plus appréciées par les employés d'*Edifixio*.

Pour poser ses jours de congé, nous possédons une application « fait maison » nommée *TRH*. Elle nous permet de déposer nos absences grâce à une vue calendaire, tout en ayant un aperçu en temps réel des congés pris et restants.

Les employés peuvent travailler sur différents projets durant la même journée. Les projets étant facturés à des taux différents, **ils doivent également pointer leur temps de travail sur l'application « *Tock* ».** Les données enregistrées sur ces fiches de temps (appelées en interne « *timesheets* ») permettent de générer de nombreux rapports. C'est très pratique pour les chefs de projet devant analyser les budgets et faire du chiffrage.

TRH est l'application ayant subi le plus de changements suite à la conduite au changement effectuée entre Atos et *Edifixio*. **Il a fallu réaliser en moins de deux semaines d'importants changements de règles métiers**, tel que l'ajout de nouvelles balances avec leur propre logique.

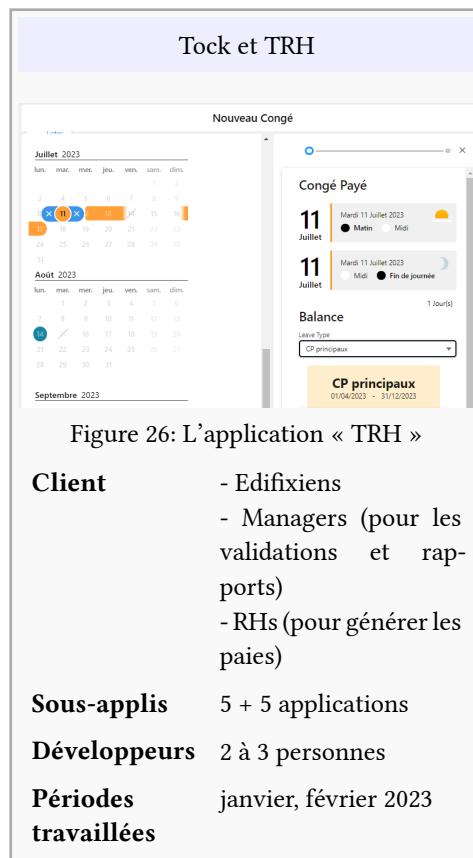


Figure 26: L'application « *TRH* »

Info

Finalement, grâce à un management bien orchestré et une *task-force*¹⁷ de sept personnes, nous avons réussi à appliquer toutes les règles d'Atos... Pour que l'outil soit finalement décommissionné trois mois plus tard !

En effet, les employés d'*Edifixio* devaient être alignés sur les nouvelles règles avant le premier avril 2023, tandis que la fermeture de l'intranet était prévue en juillet 2023 (une date qui n'était pas encore connue au moment où nous développions les correctifs).

○ Étude de cas

Chez *Edifixio*, le traitement des paies était externalisé à un prestataire de paie. En janvier, le prestataire a dû être changé vers ADP, celui utilisé par Atos. L'équipe RH devait ainsi exporter les salaires avec toutes les données requises.

Dans l'intranet, il existe déjà une application nommée « *Mass Salary Export* ». Elle fait le travail demandé, mais ne récupère pas toutes les informations requises. Le code est ancien (2016), mais surtout de très mauvaise qualité.

○ Solution mise en place

Finalement, j'ai estimé qu'une réécriture complète serait plus efficace que de devoir reprendre ce qui existait déjà. Honnêtement, ce code a été le plus mauvais que j'ai rencontré de toute ma carrière. Pour voir le bon côté des choses, il m'a incité à créer un groupe de discussion nommé « *Salesforce Horror* » (Annexe A9 (page 45)).

¹⁷Task-Force : dans le développement, création d'un groupe de développeur organisé pour la résolution de tâches urgentes

Le ticket me demande, pour chaque employé français, d'exporter :

- Les informations d'identification des employés (matricules, informations sur le contrat de travail)
- Pour chaque mois des années exportées, une colonne exportant le salaire et une colonne exportant la prime.

Il y avait deux enjeux durant le développement de cette fonctionnalité :

- Réaliser une solution dans les règles de l'art (calculs uniquement côté serveur et stricte séparation entre le front-end et le back-end, code couvert par des tests unitaires pertinents...)
- Le salaire change dans le temps. Il faut être capable de récupérer l'historique des salaires et d'assigner le bon montant pour les cellules de chaque mois. Laisser la cellule vide pour les mois dont un salarié n'a pas travaillé (arrivée ou départ en milieu d'année).

○ Résultats

Le nouveau code a été conçu pour être maintenable, testé et documenté. J'ai pu sortir la fonctionnalité dans les délais, et la migration des 320 employés s'est effectué avec succès.

III. D) L'outil de gestion de projets (*Staffing*)

Contrairement à Tock où les employés enregistrent leur temps travaillé (passé), **Staffing se concentre sur la répartition prévisionnelle** de leur journée (futur).

Sur cette application, les managers peuvent assigner aux personnes de leur équipe quel pourcentage d'une journée passer pour tel ou tel projet. Par exemple, mettre à un développeur 30% de sa journée sur un projet clientèle, 50% sur un autre et 20% sur de la préparation aux certifications.

Info

L'application est capable de gérer des cas assez précis, tel que les jours d'absences posés sur TRH. Pour plus de détails concernant la logique d'affectation de temps et de la gestion des absences, je vous redirige au rapport d'apprentissage d'Antoine Banha.



Figure 27: L'application « Staffing Grid »

| | |
|-----------------------------|------------------------------|
| Client | Managers et chefs de projets |
| Sous-applis | 4 applications |
| Développeurs | 2 à 3 personnes |
| Périodes travaillées | février à mai 2023 |

○ Étude de cas

Nous avons vu que Tock permet aux employés d'enregistrer les temps de travail. L'application est utilisée pour facturer les clients, mais aussi à générer divers rapports. Seulement, ils ne tiennent compte que du temps effectivement passé. Il n'est pas possible de faire des estimations de temps/coût dans le futur.

Les chefs de projet ont récemment exprimé le besoin d'avoir une application permettant de générer divers types de rapports. Voici les principaux usages :

- **Extraction brute de données** : récupérer les données avec très peu de traitement, pour que les manageurs fassent eux-mêmes l'analyse des données sur un tableur Excel.
- **Taux d'activité** : Pour visualiser la proportion de temps qu'un employé a passé sur chaque type de projet (clientèle, projets internes, formations, absences...). Ce rapport est aussi utilisé pour identifier les personnes ayant du temps libre.
- **Comparaison entre Tock et Staffing** : En regardant le temps prévisionnel, on compare à quel point le temps effectivement réalisé (Tock) est proche de la précédente estimation (Staffing). Un des rares rapports où la période sélectionnée sera le passé.

- **Chiffrage de projets** : Grâce aux pourcentages entrés sur l'application, on peut déterminer le coût d'un projet et affiner le résultat sur différents paramètres (temps, employés). La difficulté : chaque membre d'une équipe ne travaille pas sur les mêmes horaires, et possède son propre salaire.

Ainsi, le but est de concevoir une application pouvant réaliser 5 types de rapports, totalisant 11 variantes. Certains rapports ne servent qu'à faire de l'extraction de données, tandis que d'autres possèdent des logiques d'agrégations plus poussées.

○ Solution mise en place

J'ai eu carte blanche pour créer cette nouvelle application ainsi que son architecture. Les besoins étaient les suivants :

- Implémenter progressivement les rapports demandés par les usagers.
- Cette application serait soumise à beaucoup d'évolutions et les besoins peuvent changer : le code doit être maintenable et ouvert à l'extension.
- Certaines parties du code seront réutilisées sur d'autres applications et processus : il faut réduire les dépendances et isoler les différentes implémentations.

Info

En 5 mois, j'ai pu rédiger 64 classes Apex sur un total de 190 heures de travail¹⁸. J'accorderai dans cette section une attention accrue par rapport aux autres applications, en particulier sur l'ingénierie logicielle qui la compose.

Figure 28: L'application « Staffing Report » : plusieurs types de rapports peuvent être produits, avec différents paramètres de calcul

À l'occasion d'un déplacement à Nantes, j'ai pu demander un avis extérieur auprès d'un collègue non-utilisateur de l'application. Verdict : après quelques *effets démo*¹⁹, ce n'était « pas génial du tout ».

¹⁸190 heures : selon le rapport que j'ai généré depuis Tock, 27.2 journées humaines. Il suffit de multiplier par 7 pour donner le nombre d'heures.

¹⁹Effet démo : Lorsqu'une application jusqu'alors utilisée sans incident présente un dysfonctionnement lors d'une démonstration publique.

De mon expérience avec l'intranet, générer des rapports était à mes yeux la création automatisée de tableaux Excel. Aux yeux d'un *data analyst*²⁰, la clé du reporting est surtout de comprendre le besoin d'un client et faire ressortir visuellement les informations utiles (qui feront gagner de l'argent).

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | |
|---|-----------------------|-------------|------------|------------|----------------|------------|----------------|----------------|--------------|------------|-------------|---------------|---------|------------|----------|--------------|------------------------|
| 1 | Nom | Manager | Date début | Date fin | Statut utilisa | Commentair | Practice | Group | Profil | Facturable | Proposition | Non Facturale | Absence | Inaktivité | Indirect | Jours ouvrés | Disponibilité % factur |
| 2 | Alexandre CHABERT | Eméric BATU | 18/06/2023 | 24/06/2023 | | | Digital Platfc | Digital Platfc | Expert-Medi | 3 | 0 | 0 | 0 | 2 | 0 | 5 | 0 |
| 3 | Loïk RAIMBAUT | Eméric BATU | 18/06/2023 | 24/06/2023 | | | Digital Platfc | Digital Platfc | Expert-Senic | 0 | 0 | 0 | 1 | 1,2 | 0 | 5 | 2,8 |
| 4 | Nicolas VERGEMANUEL G | Emmanuel G | 18/06/2023 | 24/06/2023 | | | Digital Platfc | Digital Platfc | Software De | 5 | 0 | 0 | 0 | 0 | 0 | 5 | 0 |
| 5 | Widad GHOUCHBAR | Eméric BATU | 18/06/2023 | 24/06/2023 | | | Digital Platfc | Digital Platfc | Expert-Medi | 2,5 | 0 | 0 | 0 | 1 | 0 | 5 | 1,5 |

Figure 29: Rapport généré avec l'application Staffing Report

En demandant à mon responsable fonctionnel les différents usages des rapports, j'ai pris l'initiative de proposer des maquettes pour le développement d'un nouvel écran, basé sur le code de Staffing Report :

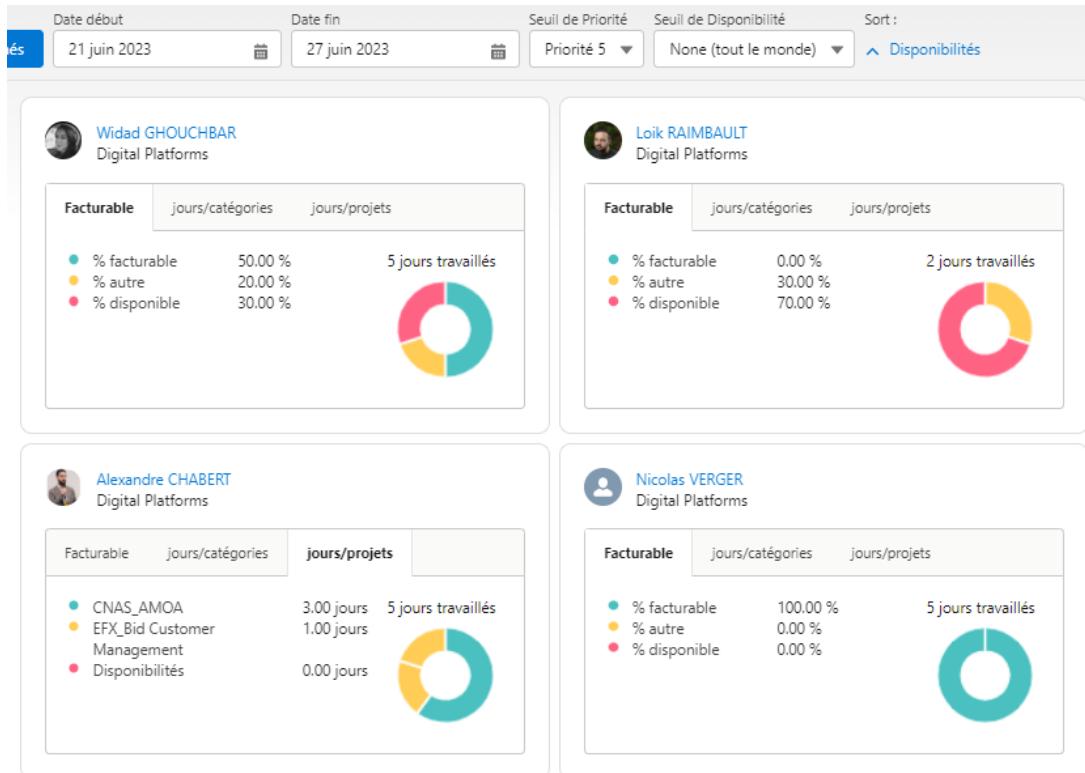


Figure 30: L'application « Activity rate ». Ce sont exactement les mêmes données que la Figure 29. Bien plus joli, non ?

En l'occurrence, le besoin est d'identifier les personnes ayant du temps libre. Pour cela, les manageurs peuvent visualiser la répartition du travail prévisionnel avec trois couleurs. La couleur rouge signifie une disponibilité (mauvais, car nécessité de « staffer »), la jaune signifie une activité non facturée (formation, POC²¹, absence, outil interne...) et le bleu signifie un projet client. Un système d'onglets permet de dévoiler des vues plus détaillées.

Pour revenir à l'application originale (Staffing Report), j'ai choisi d'employer une architecture multicouche, particulièrement adaptée à ce genre de fonctionnalité. Dans les grandes lignes, cette architecture a déjà été expliquée sur la [section II. E \(classes apex et architecture logicielle\)](#).

²⁰Ingénieur qui a pour mission « d'exploiter et interpréter les données pour en dégager des observations business utiles » [10]. Malgré ma spécialisation en ingénierie de conception logicielle, d'un point de vue fonctionnel, les missions de reporting se rapprochent davantage à ce poste.

²¹POC : Proof Of Concept, aussi nommé « bid » en anglais. Prototype servant à démontrer la faisabilité d'un projet, généralement en amont de l'accord commercial définitif.

◎ Architecture de l'application et ingénierie logicielle

Couche de présentation et abstraction des générateurs de rapport —

Les rapports de staffing partagent en commun le fait d'extraire et calculer des données sur une période de temps et ciblant soit une liste de projets, soit une liste d'utilisateurs. Ils reçoivent le même format de données en entrée et en sortie. Seule la récupération de données et la logique de calcul diffère selon le type de rapport.

On veillera ainsi à séparer la logique en briques isolées. De cette manière, on sera sûrs que la modification d'un rapport n'impactera pas d'autres, et le code logique d'un rapport sera réutilisable sur d'autres applications. En normalisant le format de données en entrée et en sortie, il sera facile d'ajouter de nouveaux types de rapports.

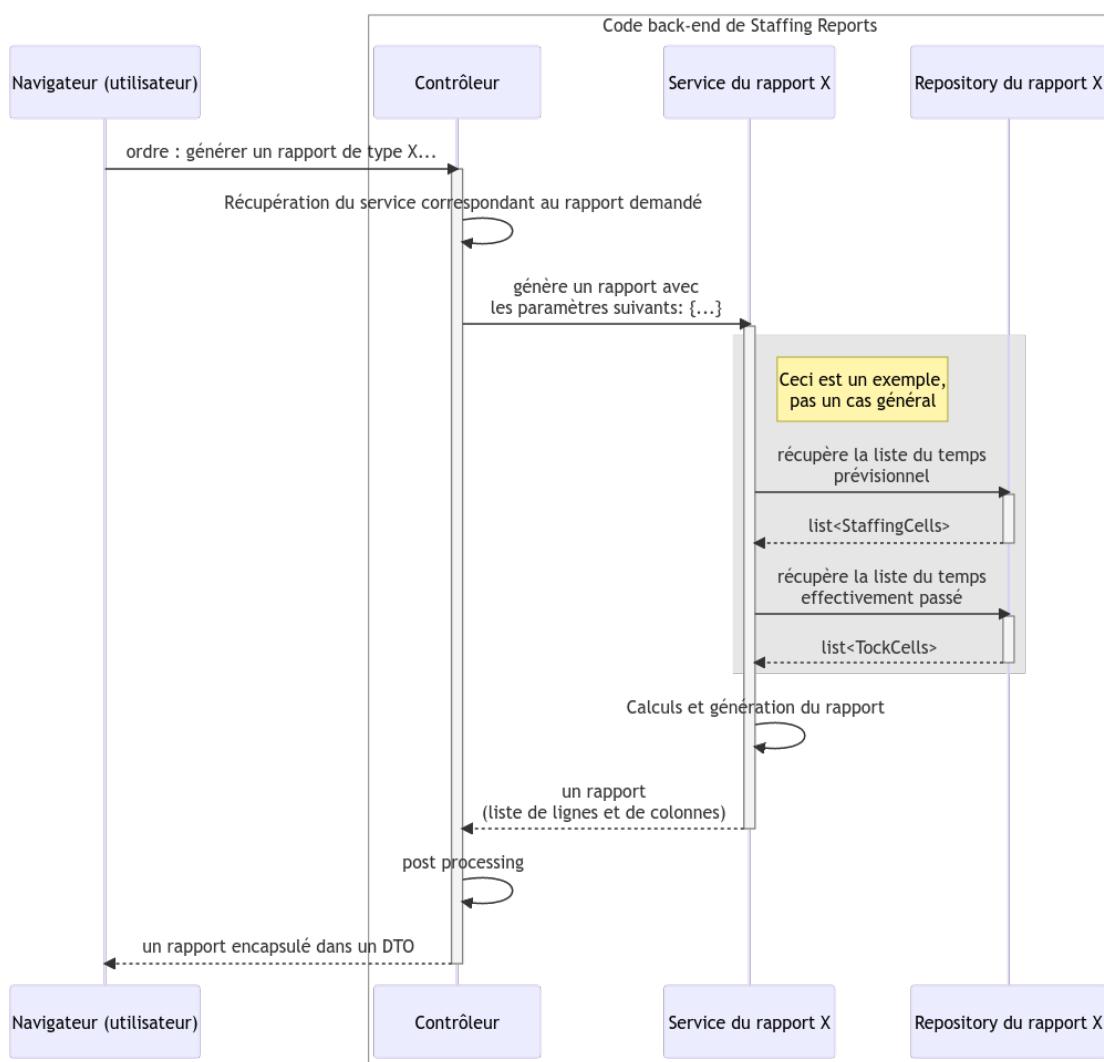


Figure 31: Diagramme de séquence récapitulant l'architecture multicouche de Staffing Reports

En pratique, voici les responsabilités de la couche présentation (contrôleur) :

- Réceptionner les requêtes venant de l'application Staffing
- Identifier le type de rapport choisi par l'utilisateur, et appeler le service qui lui est associé
- Une fois le rapport généré à partir de la classe de service, nous la formaterons dans un objet « DTO »²² de sortie

²²DTO: Data Transfer Object - Classe ne contenant que des propriétés publiques, pour faire l'intermédiaire entre deux couches. Similaire au principe de structure dans le langage C.

La couche « application » (service) contient les différents générateurs de rapports. Ils implémentent tous une interface avec une méthode générique nommée `generateReport`. Avec les paramètres du rapport reçus en entrée, tout générateur de rapport devra renvoyer une liste de colonnes et lignes constituant le tableau. Grâce à un mécanisme d'injection de dépendance, le contrôleur pourra *se brancher* sur le bon service.

Voici un diagramme de classes simplifié, représentant les deux premières couches de Staffing Reports :

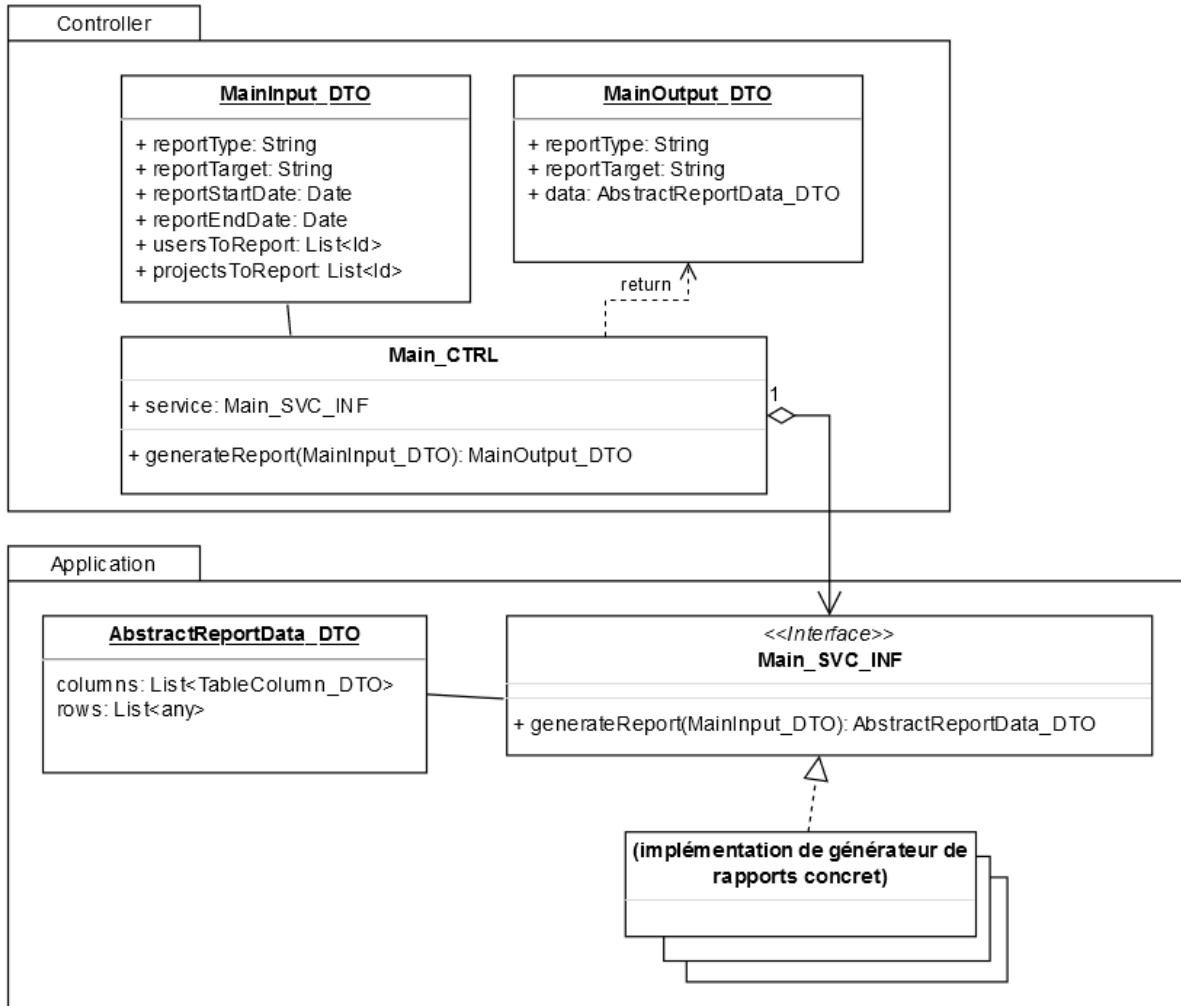


Figure 32: Diagramme de classes simplifié pour la couche de présentation

Implémentation concrète d'un générateur de rapport —

Pour créer un générateur de rapport, il faut faire une classe de service (calcul et logique de génération) qui récupérera ses données depuis la couche *repository*. Comme avec la couche de présentation, la couche application et persistance sont isolées par une interface et un mécanisme d'injection de dépendance.

Info

Additionnellement, chaque classe concrète possède une classe de test unitaire, et si nécessaire, une classe de *mock* (ou « *faker* »). La classe de *mock* est déterministe, c'est-à-dire qu'elle génère des données en dur plutôt que d'appeler une base de données. Injectée dans les tests unitaires de la couche inférieure, cela permet de s'assurer que chaque couche dépend des abstractions et non des implémentations²³.

En pratique, pour la création de l'application Activity Rate, j'ai pu créer un contrôleur différent, qui appelle le service de génération de rapport au même nom. Sur le même principe, nous avons ajouté un système de notification hebdomadaire, alternant par mail à tous les manageurs les personnes ayant un faible taux d'occupation.

Voici un diagramme de classe récapitulant l'architecture liée au rapport Activity Rate :

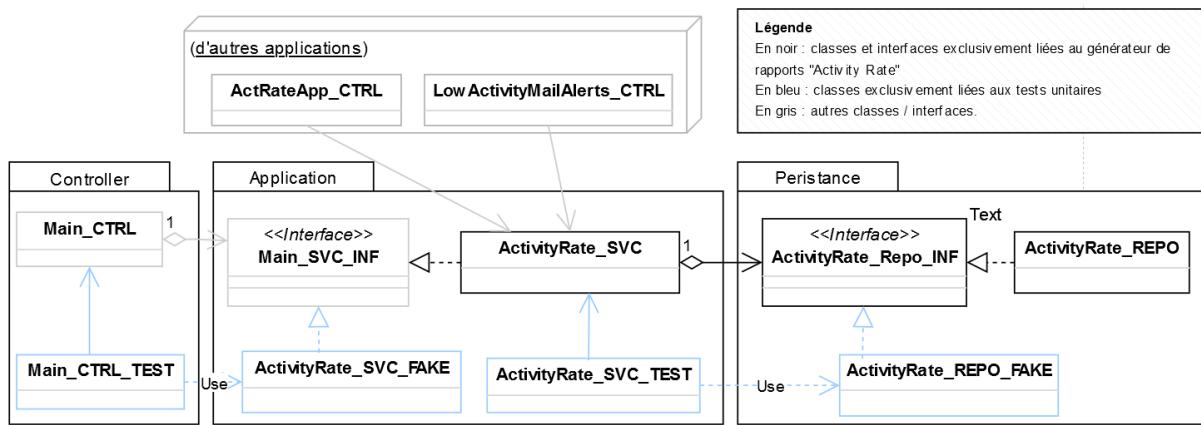


Figure 33: Diagramme de classes répertoriant tous les éléments impliqués dans la génération de rapports de type « Activity Rate ». Les classes en noir concernent le générateur de rapport concret.

²³Règle de ségrégation des interfaces dans les principes SOLID. Nous avons vu en classe ces principes, mais nous n'avons pas encore vu ce concept d'architectures en couches.

IV) Bilan

IV. A) Récapitulatif

Globalement, je tire un bilan très positif de cette année d'apprentissage. J'ai pu mettre à l'œuvre ce que j'ai appris à travers les cours, mais aussi à travers de l'expérience de mes collègues.

J'ai pu monter en compétence et renforcer mes acquis sur le développement au sein de la plateforme Salesforce, grâce aux nombreuses tâches qui m'ont été assignées (en particulier celles que j'ai pu mentionner dans mon rapport).

La conduite au changement portée par la fusion d'Edifixio en Eviden France a été assez challengeante et m'a permis de travailler sur des tâches qui m'ont fait sortir de ma zone de confort. Comparé à ma précédente expérience de stage, j'ai eu l'occasion de m'améliorer, en développant les fonctionnalités sous trois jours en moyenne (contre 5 à 7 jours auparavant) ou en prenant plus d'initiative sur la suggestion de solutions (création de l'outil de visualisation de données « Staffing Activity Rate »).

Par mes missions très variées et enrichissantes, j'ai pu comprendre les différents enjeux des activités que j'ai pu mener. L'organisation d'activités transverses comme le *Coding Dojo* et l'évènement de *Chasse aux œufs* m'a permis de rencontrer de nouvelles personnes au sein du campus et de renforcer la cohésion d'équipe.

Cette année m'a conforté dans ma volonté d'évoluer dans ce secteur, pour ainsi mettre à contribution mes acquis dans le développement.

IV. B) Difficultés rencontrées

Durant mon année d'apprentissage, mes difficultés ont été principalement d'ordre technique. Je me suis bien intégré au sein de mon équipe, et les impacts de la conduite au changement n'ont pas vraiment perturbé la bonne réalisation de mon travail.

Tout d'abord, **la nature des missions**. Salesforce n'est pas conçu pour créer un système d'informations complet. Créer un intranet avec des règles spécifiques et le maintenir dans le long terme est assez difficile. Beaucoup de code « historique » n'a pas été conçu dans les règles de l'art, à une époque où la qualité n'avait pas autant d'importance qu'à notre arrivée.

Ensuite, **l'écosystème Salesforce**. Les langages de programmation utilisés sont spécifiques à la plateforme, et évoluent très lentement. Les ressources disponibles sur internet sont faibles, et certains mécanismes « modernes » ne sont pas disponibles. Parmi eux, quelques notions vues en cours qui sont inexistantes en Salesforce : les espaces de noms, la programmation fonctionnelle, la généricité et les mécanismes de concurrence. Si elles étaient présentes, cela aurait grandement amélioré mon expérience de développement.

Enfin, **les choix de conception logicielle**. Par exemple, nous avons vu dans la dernière partie de la section III l'emploi d'une architecture multicouche pour l'application « Staffing Reports ». Cette méthode possède beaucoup d'avantages sur le long terme, mais prend beaucoup de temps à mettre en place (dans des projets clients soumis à de fortes contraintes de temps, ce n'est pas rentable).

IV. C) Grille eCF3 renseignée et datée

Identification du profil du poste : (3.2.) Concepteur / Développeur

Info

La grille (ainsi que l'auto-évaluation de chacune des compétences requises) est directement jointe sur le rendu Moodle. Je l'ai également mise à jour vis-à-vis du référentiel Cigref de l'année 2022.

Une copie est disponible sur : https://efrei365net-my.sharepoint.com/:f/g/personal/logane_tann_efrei_net/Eqr7LIuHRnpLpN32sV0Djl8B0INrg2i_TsICP0kL_xUBhA?e=fTp2ZB

Dans les grandes lignes, j'ai acquis 70% des compétences demandées selon la notation Cigref. Cependant, certains descriptifs de compétences ne peuvent s'appliquer à ma mission. Je peux citer par exemple la construction de réseaux ou l'étude de plateforme et la structuration des systèmes, car ceux-ci sont déjà gérés par la plateforme cloud monolithique de Salesforce. D'autres compétences, telles que la conception des spécifications, est relativement limitée vu que j'effectue essentiellement de la maintenance évolutive sur des applications existantes.

IV. D) Perspectives d'évolution

Aujourd'hui, en juillet 2023, l'intranet d'Edifixio a été remplacé par les outils d'Atos. Comme il serait dommage de jeter l'équivalent de cinq ans de travail, nous avons pour objectif de travailler dès maintenant sur l'*assetisation²⁴* du **Salesforce Interne**.

En parallèle, j'ai été ajouté sur un projet nommé **SandboxEs**. J'en ai rapidement parlé dans la partie I du rapport, cet outil de développement servira aux développeurs pour migrer des données cohérentes entre deux instances Salesforce. Un objectif qui semble anodin, mais qui cache beaucoup de limitations techniques.

Sandboxes n'est pas une application Salesforce comme celles que nous avons vues dans le rapport. Il s'agit d'une application complète, avec sa propre pile technique (en NodeJS). Elle utilise les APIs Salesforce, ce qui ouvre un angle de vue totalement différent à ce dont j'ai eu l'habitude de travailler.

Comme je suis le seul développeur actif assigné au projet, on me laisse beaucoup d'autonomie sur les choix de conception et l'organisation de mes tâches. **Ce sera donc une bonne occasion pour compéter les compétences du référentiel Cigref** dont je n'ai pas eu l'occasion de développer:

Intégration des systèmes

SandboxEs est une application conçue avec le framework front-end Next.js (React) et back-end Nest.js. Elle utilise aussi la base de données PostgreSQL, ainsi que Redis pour mettre en cache les données des tâches exécutées en parallèle. L'application utilise beaucoup de dépendances, telles que RxJs ou MaterialUi. C'est différent de Salesforce, qui possède une approche propriétaire et monolithique. Désormais, l'interopérabilité des différents modules ne se fera pas à notre place.

Production de la documentation

Sur l'intranet, seules les documentations techniques étaient rédigées par les développeurs. Les spécifications fonctionnelles étaient simplement dites à l'oral ou bien rédigées par notre chef de projet. Sur SandboxEs, seules quelques consignes et un objectif final est donné. Je devrais rédiger les spécifications fonctionnelles par moi-même, et être force de proposition.

Ingénierie des systèmes TIC

Étant donné la pile technique assez lourde de SandboxEs, qui se rapproche d'une architecture microservices, il faudra faire le lien entre les différents systèmes. Je serai également responsable des choix de l'infrastructure (comparer les offres cloud, s'assurer de l'efficacité du modèle choisi...)

Étude de gestion des risques

SandboxEs sera utilisé par des clients, leurs données seront entre nos mains. Pour qu'ils participent à la *beta* de cette application, il faudrait que les risques soient évalués et que tout soit fait pour maximiser la sécurité.

Le programme de l'année prochaine sera particulièrement utile pour la création de SandboxEs. Nous verrons la sécurité des systèmes, l'intégration et le déploiement continu, les systèmes en temps réel et des concepts avancés en programmation orientée objet (avec C#).

²⁴ Assetisation : Jargon technique désignant le fait de transformer un code en tant que solution isolée et potentiellement réutilisable / vendable (un « asset »)

Conclusion

Pour conclure, j'ai effectué ma première année d'apprentissage en tant qu'ingénieur logiciel Salesforce, au sein de l'entreprise Edifixio (désormais Eviden France).

Mes activités portaient essentiellement sur le développement de l'intranet d'Edifixio, le « Salesforce Interne ». La fusion accélérée d'Edifixio à Eviden France signifiant une conduite au changement et une décommission très proche de notre intranet (juillet 2023), nos tâches ainsi que leurs priorités ont fortement été impactées cette année.

Tout d'abord, j'ai été assigné à l'export des factures de l'application de comptabilité Achats. Ensuite, à la modification de l'application de poses de congés, pour appliquer les nouvelles règles d'absences. Puis, à l'extraction des salaires stockés sur l'application RH, pour le changement du prestataire de paie. Enfin, à la création du module de reporting pour l'application Staffing, application dont nous espérions garder en vie malgré l'abandon du Salesforce Interne.

Mon expérience ne se résume pas uniquement à la simple exécution de ces missions. Cette année était très enrichissante pour deux raisons : d'une part, la vie d'entreprise au sein d'une jeune société de services numériques, et d'autre part, découvrir tout un univers à propos des logiciels de gestion de la relation client grâce à la maintenance d'applications Salesforce. Des expériences qui n'auraient pas été possibles sans avoir pu suivre un cursus d'apprentissage à l'EFREI.

En effet, j'ai pu appliquer et approfondir mes connaissances en architecture logicielle, en création d'interfaces graphiques et en méthodologies agiles. Durant deux mois, je me suis également efforcé à respecter le « Serment du Programmeur » [11] de Robert C. Martin; en particulier que « le code produit sera toujours issu de mon meilleur travail » et que je « ne rendrai jamais mon code plus mauvais que le précédent ».

En tant que passionné de programmation, une règle se démarque parmi toutes : « Je n'arrêterai jamais d'apprendre et de m'améliorer dans mon métier ». Bien que l'intranet ne soit plus utilisé aujourd'hui, il me reste encore plein d'opportunités pour évoluer. Actuellement, je suis aussi développeur principal d'un futur outil interne, nommé SandboxEs. Ce sera probablement mon sujet de mémoire pour l'année prochaine, et je pourrai gagner en autonomie et en compétences sur la conception de cette application.

Fort de cette expérience enrichissante et diversifiée au sein d'Edifixio, je projette de rester dans l'entreprise après mes études. Toutefois, je crains que la transformation de l'entreprise en une plus grande structure dépersonnalise progressivement l'esprit porté par les employés historiques d'Edifixio. Il me reste trois ans pour me décider si cette projection sera mon choix final, alors j'ai encore beaucoup de temps pour réfléchir à mon futur.

Table des figures

| | |
|--|----|
| Figure 1: Logo d'Edifixio | 7 |
| Figure 2: Dates clés d'Edifixio | 7 |
| Figure 3: Logo d'Atos et d'Eviden | 7 |
| Figure 4: Logos des 4 plateformes cloud maîtrisées | 8 |
| Figure 5: Organigramme fonctionnel | 9 |
| Figure 6: Photographie à l'extérieur et à l'intérieur des bureaux Atos | 9 |
| Figure 7: Photographie de mon espace de travail | 9 |
| Figure 8: De gauche à droite : L'application « Calendrier de l'avent Edifixien » et « Chasse aux œufs Edifixienne » | 10 |
| Figure 9: Essai de l'application <i>pâques</i> sur un vrai QR code | 10 |
| Figure 10: Une session de Coding Dojo | 11 |
| Figure 11: Une session de Knowledge Sharing | 12 |
| Figure 12: Photographie de l'équipe Salesforce au musée Rodin | 12 |
| Figure 13: Atelier interactif de cadrage de projet avec l'équipe AWS | 12 |
| Figure 14: Différentes fonctionnalités d'un CRM | 14 |
| Figure 15: Capture d'écran de l'interface Salesforce | 14 |
| Figure 16: Aperçu de l'application « Tock », où je déclare la répartition de mon temps travaillé durant la semaine en cours. | 15 |
| Figure 17: Captures d'écran respectives de la documentation interne sur Confluence et du tableau Kanban sur Jira | 16 |
| Figure 18: Cycle de vie d'un ticket et branches GitLab | 17 |
| Figure 19: Un exemple de code Apex avec une requête SOQL et une instruction DML | 18 |
| Figure 20: Un exemple de flow Salesforce | 19 |
| Figure 21: Interface de ChangEs v2, pour sélectionner les classes à déployer en production | 19 |
| Figure 22: L'arborescence du projet, avec une classe de l'application Staffing sélectionnée | 20 |
| Figure 23: Domain pattern en trois couches (contrôleur, service et repository) | 21 |
| Figure 24: L'application « Bill management » | 23 |
| Figure 25: Diagramme de séquence simplifié représentant la solution choisie | 24 |
| Figure 26: L'application « TRH » | 25 |
| Figure 27: L'application « Staffing Grid » | 26 |
| Figure 28: L'application « Staffing Report » : plusieurs types de rapports peuvent être produits, avec différents paramètres de calcul | 27 |
| Figure 29: Rapport généré avec l'application Staffing Report | 28 |
| Figure 30: L'application « Activity rate ». Ce sont exactement les mêmes données que la Figure 29. Bien plus joli, non ? | 28 |
| Figure 31: Diagramme de séquence récapitulant l'architecture multicouche de Staffing Reports | 29 |
| Figure 32: Diagramme de classes simplifié pour la couche de présentation | 30 |
| Figure 33: Diagramme de classes répertoriant tous les éléments impliqués dans la génération de rapports de type « Activity Rate ». Les classes en noir concernent le générateur de rapport concret. | 31 |
| Figure 34: Localisation des locaux Atos en périphérie parisienne | 39 |
| Figure 35: Un exemple de code commentaire de revue de qualité de code | 39 |
| Figure 36: Un exemple de code Javascript représentatif d'un composant LWC | 40 |
| Figure 37: Une fonction mathématique. Une entrée = une sortie attendue, tout simplement. | 41 |
| Figure 38: Un test unitaire avec le classique exercice du « FizzBuzz », structuré en AAA | 41 |
| Figure 39: Un exemple de documentation technique, avec le composant « headless tabset » | 42 |

| | |
|---|----|
| Figure 40: Extrait de la documentation technique du back-end de l'application Validation Center ... | 44 |
| Figure 41: Aperçu du groupe de discussion « Salesforce Horror » | 45 |

Info

L'ensemble des figures proviennent de moi-même ou de documents internes, sauf présence de mention « source » à l'emplacement de l'illustration.

Bibliographie

- [1] Edifixio, “Présentation D'edifixio.” <https://www.edifixio.com/en/edifixio>
- [2] Marion Delmas, “Atos finalise l'acquisition d'edifixio.” https://atos.net/fr/2020/communiques-de-presse_2020_12_01/atos-finalise-lacquisition-dedifixio
- [3] Edifixio, “Salesforce | edifixio.” <https://www.edifixio.com/fr/salesforce>
- [4] Dan Runkevicius, “How amazon quietly powers the internet.” <https://www.forbes.com/sites/danrunkevicius/2020/09/03/how-amazon-quietly-powers-the-internet/>
- [5] Kaizen Developer, “Approches différentes de coding dojo.” <https://kaizendevoloper.wordpress.com/2020/03/27/approches-differentes-de-coding-dojo/>
- [6] Salesforce, “Qu'est-ce que salesforce ?” <https://www.salesforce.com/fr/products/what-is-salesforce/>
- [7] Cogivea CRM, “Qu'est-ce qu'un logiciel CRM ? Définition.” <https://www.cogivea.com/ressources/faq-questions-frequentes/87-generalites-sur-le-crm/51-qu'est-ce-qu-un-logiciel-crm-definition.html>
- [8] (Collectif), “Page wikipédia “salesforce”.” <https://fr.wikipedia.org/wiki/Salesforce>
- [9] (Collectif), “Les trois couches - architecture trois tiers | wikipedia.” [https://fr.wikipedia.org/wiki/Architecture_trois_tiers#Couche_de_pr%C3%A9sentation_\(premier_niveau\)](https://fr.wikipedia.org/wiki/Architecture_trois_tiers#Couche_de_pr%C3%A9sentation_(premier_niveau))
- [10] Michael Page, “Fiche métier : Data analyst.” <https://www.michaelpage.fr/advice/metiers/digital-marketing-communication/fiche-m%C3%A9tier-data-analyst>
- [11] Robert C. Martin, “The programmer's oath. .” <https://blog.cleancoder.com/uncle-bob/2015/11/18/TheProgrammersOath.html>

[A] Annexes

Table des annexes

| | |
|--|----|
| [A1] Localisation des locaux d'Edifixio | 39 |
| [A2] Exemple de revue de qualité de code | 39 |
| [A3] Exemple d'un composant LWC | 40 |
| [A4] Support de la langue anglaise | 40 |
| [A5] Tests unitaires, tests sur Salesforce | 41 |
| [A6] Exemple de documentation technique d'un composant front-end | 42 |
| [A7] Gestion de projet en « Kanban » et versionnage de code | 43 |
| [A8] Architecture logicielle de l'application Validation Center | 44 |
| [A9] Exemple du réseau social d'entreprise, Chatter | 45 |

[A1] Localisation des locaux d'Edifixio

Les locaux Atos sont situés à proximité du Pont de Bezons (terminus du T2). À vol d'oiseau, ils sont situés à 4 km de La Défense et à 11 km du centre de Paris.



Figure 34: Localisation des locaux Atos en périphérie parisienne

En comparaison, l'Efrei est situé à 8 km du centre de Paris. Étrangement, la plupart des locaux d'Atos sont collés à une frontière de banlieue, à une heure d'un centre-ville. Du moins, c'est le cas pour les pôles de Nantes, Lyon et Grenoble. Edifixio a souffert d'une légère perte d'attractivité à l'emploi, à cause des déménagements plus lointains.

[A2] Exemple de revue de qualité de code

Lorsque nous souhaitons fusionner notre code sur la branche principale, nous devons assigner un « mainteneur » pour qu'il fasse une revue de qualité. Il devra lire le code que nous avons produit et signaler un code trop difficile à comprendre, un problème de sécurité ou le non-respect d'une de nos conventions de codage.

À la détection d'un problème, le mainteneur est invité à déposer un commentaire, ce qui va créer un fil de discussion. Le développeur doit argumenter sur ses choix et corriger si nécessaire le problème. Une fois que le mainteneur valide la correction, il peut marquer le fil de discussion en tant que « résolu ».

Lorsque tous les fils sont résolus, le mainteneur effectue une dernière passe et procède à la fusion du code.

```

68 +     async initChartOnNextLwcTick() {
69 +       await Promise.resolve();

```

Daniel AGUIAR @daniel.laguiar · 2 months ago

Why do you need to resolve an empty promise here ?

1 Reply

Logan TANN @LoganTann · 2 months ago

That's the standard way to run asynchronous code on the next LWC tick (once the component and its DOM is fully initialized).

As this method may run before painting (during connectedCallback), if I remove this statement, the QuerySelector on the line below will fail.

- Logan TANN changed this line in version 3 of the diff 2 months ago · [Compare changes](#)

Figure 35: Un exemple de code commentaire de revue de qualité de code

[A3] Exemple d'un composant LWC

The screenshot shows a code editor with two files: `Code JavaScript` and `template.html`.

Code JavaScript:

```

/*
 * Created by LoganTann <logan.tann@edifixio.com> on 09/05/2022
 *
 * This is just a demo component for the internship report.
 */

import { api, LightningElement, track, wire } from "lwc";
import getDemoData from "@salesforce/apex/CORE_DemoClass_CTRL.getDemoData"; ①

export default class demoComponent extends LightningElement {
    /**
     * The preset to fetch
     * @type {string}
     * @see #getDemoData
     */
    @api presetLabel; ②

    /**
     * The data returned by the controller
     * @type {[{year: string, data: string}}]
     */
    @track demoData = []; ③

    get processedDemoData() {
        return this.demoData.map((column, i) => ({
            data: column.data,
            isActive: column.year === "2022",
            key: i,
        }));
    }

    // ---- data fetching ----
    /**
     * @type {lwc.WireInfo|null}
     */
    wiredData = null;
    @wire(getDemoData, { preset: "$presetLabel" })
    getDemoData(wireInfo) {
        this.wiredData = wireInfo;
        this.demoData = JSON.parse(wireInfo.data);
    }
    handleRefresh() {
        this.refreshApex(this.wiredData);
    }
}

<template>
    <button onclick=>handleRefresh</button>
    <template for:each={processedDemoData} for:item="line">
        <p key={line.key} if:true={line.isActive}>{line.Data}</p>
    </template>
</template>

```

Annotations:

- On pourra appeler un contrôleur Apex grâce à un import généré dynamiquement, au lieu de devoir retenir une URL compliquée.
- Les commentaires JsDoc sont pratiques pour documenter notre code, mais sont aussi utiles pour typé notre code (une migration complète vers typescript étant trop difficile, JsDoc offre les bénéfices du typage statique tout en assurant une rétrocompatibilité)
- Une variable avec le décorateur `@api` signale que le composant prends un paramètre via un attribut HTML. Avec le décorateur `@track`, signale au composant de recharger son rendu lors d'une mutation de la propriété
- Une fonction avec le décorateur `@wire` permet de faire une requête sur un contrôleur Apex. L'objet retourné est utilisé pour le chargement ou l'envoi des données (avec la méthode `refreshApex`)
- Interdiction d'évaluer dynamiquement dans le template html, pour des bien choisies par Salesforce. À la place, on doit se repérer sur des getters JavaScript (variables dynamiquement calculées)

template.html:

Figure 36: Un exemple de code Javascript représentatif d'un composant LWC

[A4] Support de la langue anglaise

Edifixio est une entreprise française ayant des implantations à l'étranger (tel qu'en Inde ou en Tunisie). La plupart des applications doivent donc supporter deux langues : le français et l'anglais.

En développement logiciel, nous utilisons des « labels » pour gérer les traductions. Les labels sont des variables que nous pouvons importer dans notre code. Nous définissons la valeur anglaise et française de cette variable, puis durant l'exécution, la bonne variante sera lue.

Salesforce intègre directement des fonctionnalités permettant de créer les labels et leurs traductions. Malheureusement, l'interface est très peu efficace, car il faut au moins 10 clics pour créer un label, sans compter le remplissage des champs du formulaire. Cela décourage souvent à les créer, mais en interne, nous sommes obligés de ne pas laisser du texte français en dur.

[A5] Tests unitaires, tests sur Salesforce

Créer un programme, c'est écrire de nombreuses fonctions. En mathématiques, une fonction n'est qu'un processus qui, donnée des paramètres d'entrée, donnera un certain résultat (ou effet de bord, selon le paradigme de programmation). La bonne pratique est de faire en sorte qu'une fonction ne fasse qu'une seule chose.

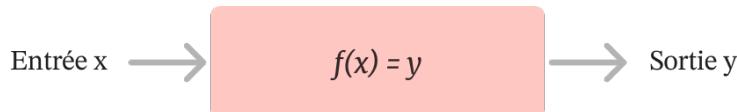


Figure 37: Une fonction mathématique. Une entrée = une sortie attendue, tout simplement.

Lorsque nous faisons du développement, nous avons plusieurs problématiques. Tout d'abord, dans le cas d'un bug à corriger, **comment prouver au client que notre implémentation le corrige bien ?** Il y a également le classique : 1 bug corrigé, 2 nouveaux bugs créés. **Comment prouver que notre code n'a pas provoqué des régressions ?** Nous pourrions tester à la main chaque fonctionnalité, mais ce n'est pas du tout viable.

La solution, c'est d'employer des tests unitaires. C'est une suite de programmes qui vont vérifier que de petites unités de notre code donnent le résultat attendu donné une entrée bien définie.

Dans l'exemple ci-dessous, nous avons une méthode de test avec trois parties :

- Given (ou Arrange) : On initialise les données d'entrées. Cela peut être des variables, mais aussi l'insertion d'objets dans la base de données.
- When (ou Act) : On exécute l'unité de code que l'on souhaite tester. Sauf impossibilité, on ne teste qu'une seule fonction/méthode/procédure.
- Then (ou Assert) : On vérifie que le résultat correspond bien à ce que nous attendons. Dans le cas où la fonction ne retourne rien, on vérifiera les effets de bords qu'elle a produits (ex : insertion d'un objet dans la base de données ?).

```

    @Test public void testFizzbuzz_shouldReturnFizzBuzz_whenMultipleThreeAndFive() {
        // Given
        int x = 15;
        // When
        String y = fizzbuzz(x);
        // Then
        Assert.assertEquals("FizzBuzz", y);
    }
  
```

Figure 38: Un test unitaire avec le classique exercice du « FizzBuzz », structuré en AAA

Il y a une dernière étape, non décrite, qui est le cleanup. Chaque test devant être indépendant, il faut veiller à réinitialiser la base de données avant chaque exécution. Fort heureusement, Salesforce gère cette étape pour nous.

Info

D'ailleurs, dire que les tests sur Salesforce sont des tests unitaires est un abus de langage. Comme la base de données peut être utilisée, il s'agit de tests d'intégration. Vu qu'un test d'intégration est un ajout par-dessus le concept de tests unitaire, il demeure possible de faire de l'unitaire pur avec des mocks et des fakers, facilement réalisable lorsque nous adoptons une architecture 3-tier.

Sur Salesforce, 70% du code doit être couvert avant de pouvoir être envoyé en production. Cela force les développeurs à adopter les bonnes pratiques. Au fur et à mesure que la base de code grandit, les tests unitaires permettent de s'assurer que nous n'avons pas cassé les implémentations précédentes.

[A6] Exemple de documentation technique d'un composant front-end

Headless tabset

Créateur : Logan TANN
Dernière mise à jour : le déc. 16, 2022 • Vu par 3 personnes

The EfxHeadlessTabset (`c-efx-headless-tabset`) is a component similar to the `lightning-tabset`, with the only difference that it does not have a navigation bar.

This allows you to create a custom menu, and even a navigation system (see : [Tutorial - Multiple-steps modals using directional navigation and the headless tabset component](#)).

Êtes vous intelligent ?

Non Oui

A headless tabset with buttons controlling the active tab (see below example code)

Example code

```

1 <c-efx-headless-tabset active-tab-value={activeTabValue}>
2   <lightning-tab value="page1">
3     <h5 class="slds-text-heading_medium slds-m-vertical_medium">Êtes vous intelligent ?</h5>
4     <lightning-button label="Non" onclick={handleOpenTab} data-opens="page2" class="slds-m-top_xx-small"></lightning-button>
5     <lightning-button label="Oui" onclick={handleOpenTab} data-opens="page1bis" class="slds-m-top_xx-small"></lightning-button>
6   </lightning-tab>
7   <lightning-tab value="page1bis">
8     <h5 class="slds-text-heading_medium slds-m-vertical_medium">Sûr ?</h5>
9     <lightning-button label="Non" onclick={handleOpenTab} data-opens="page2" class="slds-m-top_xx-small"></lightning-button>
10    <lightning-button label="Certain" onclick={handleOpenTab} data-opens="page1" class="slds-m-top_xx-small"></lightning-button>
11  </lightning-tab>
12  <lightning-tab value="page2">
13    <h5 class="slds-text-heading_medium">Je le savais.</h5>
14  </lightning-tab>
15 </c-efx-headless-tabset>
```

```

1 export default class DemoHeadlessTabset extends LightningElement {
2   activeTabValue = "page1";
3
4   handleOpenTab(event) {
5     const clickedButton = event.currentTarget;
6     this.activeTabValue = clickedButton.dataset.opens;
7   }
8 }
```

Attributes

| Attribute | Description |
|---|---|
| Name : <code>active-tab-value</code> (Required) | The value in which we show the tab having the matching "value" attribute. |
| Slot : (default) (Required) | Multiple <code>lightning-tab</code> components with the <code>value</code> attribute set. |

Implementation Notes :

- If the `active-tab-value` attribute is invalid, this component won't log an error. It's your responsibility to make sure of the input validity.

Figure 39: Un exemple de documentation technique, avec le composant « headless tabset »

[A7] Gestion de projet en « Kanban » et versionnage de code

Le management du *Salesforce Interne* est basé sur la méthodologie « Kanban », ajouté à quelques concepts repris des *méthodologies Agile*²⁵.

◎ Principe du Kanban

Le projet se construit via de petites mises à jour à cycle régulier. On ne code pas une application complète : **on commencera par programmer les fonctionnalités minimales. Puis, on ajoutera des améliorations au fur et à mesure**, pour s'adapter dans le cas où le besoin changerait. Le chef de projet crée et assigne de petites tâches sur Jira, un *tableau Kanban* virtuel, qui suivra un cycle de vie bien défini. Ces courtes tâches sont réalisables sous 4 jours en moyenne.

Un *tableau Kanban* est un tableau avec habituellement trois colonnes de statut (To-Do, In Progress, Done), auquel on collera de courtes tâches écrites sur un post-it. **On appelle ces tâches des « tickets** », j'emploierai régulièrement ce terme tout au long du rapport. Lorsque l'état d'une tâche change, on déplace le post-it sur la colonne suivante. C'est une méthode très pratique pour représenter visuellement le progrès des tâches.

Sur notre projet, nous avons trois colonnes (= processus) de plus :

- Review (Revue de code) : Une fois la tâche finie, un collègue plus expérimenté doit relire mon code pour s'assurer de sa qualité. Un code lisible et maintenable est de loin plus important qu'une fonctionnalité livrée rapidement.
- QA (Assurance Qualité) : Maintenant que le code est revu, il faut que quelqu'un vérifie que notre implémentation respecte les spécifications fonctionnelles.
- Validé : Ces deux processus de vérification passés, je suis autorisé à migrer mon code sur l'instance de production, utilisée par les clients finaux. Une fois l'opération de mise à jour terminée, mon ticket peut passer dans la colonne « Done ».

◎ Versionnage et collaboration sur le code

Le *Salesforce Interne*, c'est environ 4000 fichiers de code utile et sept personnes travaillant à des jours différents. Pour collaborer en sécurité, il est impossible d'éditer son code à la volée et risquer d'écraser le code non terminé d'un collègue absent. Avoir son propre code sur une clé USB et copier les fichiers modifiés à la main peut aussi être fastidieux et non fiable.

Pour résoudre ce problème, on emploie un logiciel de gestion de version (le plus populaire étant Git) et un service cloud pour stocker notre code (GitLab chez nous). Il va s'occuper de gérer un historique du code pour revenir en arrière en cas de problème, et possède également d'autres fonctions, tel qu'un système de branches. **Créer une branche, c'est faire une copie du code à un instant T, et pouvoir travailler dans son coin sans pouvoir impacter d'autres personnes.**

Une fois les modifications faites sur notre branche, il faut pouvoir réintégrer le code que nous avons créé au sein du code source principal de référence. Pour cela, git peut fusionner le code pour nous. Un tel processus est appelé « merge ».

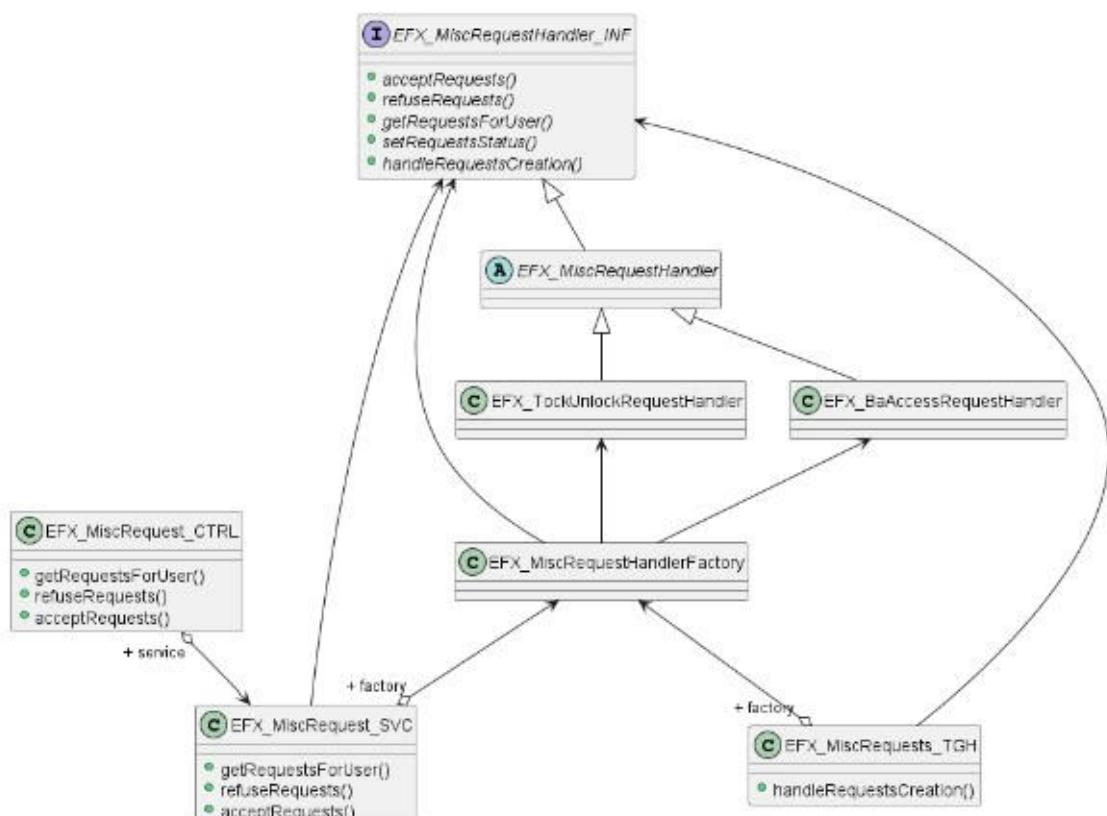
Nous l'avons vu tout à l'heure, une étape de vérification de qualité de code est obligatoire avant de fusionner notre code sur la branche principale. **Sur GitLab, on peut conditionner la fusion par une validation grâce au système de merge request.**

²⁵ Agile : Mouvement promouvant une approche basée sur l'adaptation aux changements et l'amélioration continue. Pour cela, plutôt que de développer sur des mois une fonction complète, le cycle de développement sera par petits bouts de manière itérative et incrémentale. Les aspects humains tels que l'organisation de réunions et d'activités sont également mises en avant. Il existe différentes méthodologies, telles que le « scrum » ou l'"Extreme programming"

[A8] Architecture logicielle de l'application Validation Center

Voici un extrait de la documentation technique de Validation Center. Une classe principale traite les requêtes de demande. Une interface possède des méthodes servant à traiter les actions de récupération de données, d'acceptation et de refus de requêtes.

Backend



Backend architecture as of May 2022.

The backend is architected with a hierarchical decomposition of concern, handlers do the work of processing requests (fetching, refusing, accepting). The system is open to extension, but closed to modification.

i The above diagram is not up-to-date. As of August 2022 :

- Added : Classes related to Skills Validation or Bills Approval
- Added : A public method `handleBeforeUpdate` in the trigger handler.
It is currently not open to extension, and is used to automatically fill the Deleted Date_c field when the status changes from Open to something else
- Added : A scheduled batch that will delete all unuseful records (3 months after the status of the misc request changes - ie. when Deleted Date is in the past)

Figure 40: Extrait de la documentation technique du back-end de l'application Validation Center

[A9] Exemple du réseau social d'entreprise, Chatter

Chatter est le réseau social d'entreprise servant à communiquer des informations RH, tout comme faire des publications plus informelles.

Par exemple, le groupe « Salesforce Horror », où nous partageons nos pires expériences dans des projets Salesforce (ce que soit du code incroyablement mauvais, ou de mauvaises expériences sur l'interface graphique).

The screenshot shows the Salesforce Chatter interface. At the top, there's a navigation bar with links like Social, Chatter, Groupes, Fichiers, Personnes, Rapports, and Idées. On the left, a sidebar lists 'Mes abonnements', 'Mes éléments', 'Mes favoris', 'Présentation de la société', 'FLUX', 'GROUPES RÉCENTS', and several groups including 'Salesforce Horror' which is currently selected. The main area displays a group page for 'Salesforce Horror' (Public). A post from 'Logane TANN' is shown, with the text: 'Vous ne rêvez pas, une formula salesforce de plusieurs lignes'. Below the text is a complex formula block. The post has 20 views and was shared by Terry CLARAC, Lucas DA SILVA et 4 autres. To the right, there's a 'Détails du groupe' sidebar with sections for 'Group Details' (Description: 'Groupe chatter où on partage nos pires expériences dans des projets salesforce'), 'Informations' (Share your best Salesforce gems. Bad code, bad experience...), and 'Propriétaire' (Logane TANN). At the bottom, there's a 'Gérer les membres' section.

Figure 41: Aperçu du groupe de discussion « Salesforce Horror »

Historique des révisions

3. 01 juillet 2023 - Application du feedback du tuteur enseignant
 - Réécriture complète de la fin de la partie 2, pour simplification des explications (II.C « Organisation du travail à réaliser » et II.D « Comment développe-t-on sur Salesforce »).
 - Amélioration du bilan, en particulier les perspectives d'améliorations. Modifications mineures sur les annexes.
2. 30 juin 2023 - Dépôt du rapport sur Moodle.
 - Ajout des annexes et du bilan
 - Relecture générale.
1. 23 juin 2023 - Rapport terminé, sauf annexes et bilan.



Rapport élaboré par Logan TANN le 01 juillet 2023 (révision 3)