Tailored Architecture Framework

Sommaire :

Introduction

État des lieux

L’entreprise LAE (Les Assureurs Engagés) est une entreprise d’assurance spécialisée dans les assurances-vie. Son engagement repose sur la satisfaction des clients, via son service client et sur la sécurité des données, notamment bancaires et à caractère privé.

Dans le cadre de l’évaluation d’un contrat, l’entreprise recueille des données sensibles tel que les maladies dont souffre le client, ses antécédent judiciaires, et son addiction à la cigarette et/ou à l’alcool.

L’entreprise est divisé en quatre services distinct :

- Le service client, qui est en communications direct avec les clients pour redirigé toute demande de leurs part vers le bon services ou pour modifier des données sur le client dans les bases de données de client/facturation/vente. Il peut également demander au service légal d’apporter une modification à la base de données légal via un appel téléphonique ou un mail.

- Le service de vente, qui a pour fonction de créer des plan d’action dont la réalisation consiste à sélectionner des clients et de leurs proposes des produits adaptés qui, s’ils sont intéressé déclenche la rédaction d’un contrat par le service légal.

- Le service légal, quant à lui, va enquêter sur les clients afin de déterminé les termes du contrat, ainsi que ses risques, puis rédiger ce derniers, et cela suite à une demande du service de vente.

- Le service de facturation. Il intervient à la suite de la signature d’un contrat, afin de rédiger une facture et de l’envoyer au client.

Chacun des services présentés ci-dessus utilise une application unique, dont les technologies à leurs origines sont toutes différentes, ce qui complexifie la maintenance du système dans sa globalité. Les bases de données sont également séparées, néanmoins elles ne sont pas hébergé au même endroit. En effet la base de donnée légal est stocké sur un serveur local. Bien que la sécurité de cette dernière est primordial et qu’elle ne peut être la cible d’attaque extérieures, son hébergement apporte plusieurs contrainte :

- La durée des processus est impacté : Lorsqu’un opérateur du service client doit apporté un changement à la base de données légal, alors il doit transmettre sa demande par téléphone ou mail au service légal étant donné que ce service est le seul à avoir accès à ladite base de données.
- La sauvegarde est risqué : La sauvegarde doit être effectué toutes les semaines, mais elle est réalisé à la main par le responsable informatique de l’entreprise uniquement, sur un autre serveur local auquel seul lui a accès. Cela implique que lorsque ce derniers est indisponible, aucune sauvegarde n’est réalisé et les anciennes sauvegardes ne sont pas accessible non plus.

Le microsoft access quant à lui recueil l’historique des appels et des mails des clients consigné par les opérateurs du service client. Cependant, il est accessible sans réel vérification et consiste en un répertoire partagé synchronisé entre tous les opérateurs.

Modification de l’architecture existante

Framework d’architecture

La première différence notable avec l’architecture d’origine (figure 1 de l’annexe) est dans le fait qu’il n’y a plus qu’une seule application pour les différents services. Lors de l’utilisation de l’application, l’utilisateur sera invité à se connecter, ce qui permettra de l’identifier et de fournir les vues/accès auxquelles il a le droit.

La deuxième est que nous optons pour une architecture hybride micro-service/événementiel (figure 2 de l’annexe). Dans le but de réduire les délais de traitements ainsi que les erreurs de transmission de données entre les services, un bus d’événement sera placé entre l’interface utilisateur et les bases de données. L’interface utilisateur demandera à un service de produire un événement indiquant quelle donnée(s) doit être changer en précisant des étiquettes correspondantes aux bases de données touchées par cet événement. Une fois l’événement produit et envoyé dans le bus, des services consommateurs reliés à chaque base de données vérifierons de l’étiquette leur correspondant, auquel cas ils la consommeront puis appliqueront le changements à leur base de données. Avec un tel système, nous nous assurons d’une transmission d’information de qualité ainsi que de bases de données synchronisé.
Les changement concernant la base de données légal aura cependant besoin de la confirmation du service légal afin d’être validé.

La troisième modifications apporté concerne la sauvegarde de la base de données légal effectué à la main jusqu’à lors par le responsable informatique. Un service sera créer pour effectuer un dump de la base de données de manières périodique afin de fiabiliser ce processus. Par ailleurs ce service pourrait être étendue aux autres bases de données si le souhait en est émis.

Composants et interactions

Le point d’entrée de la nouvelle architecture sera une interface homme machine, hébergé sur un serveur, qui demandera à l’utilisateur de se connecter à son compte via une session qui lui sera distribué. Une fois la connexion effectuer l’interface chargera les vues correspondant aux accès/besoins de l’utilisateur. Si ce derniers a besoin d’effectuer une modifications à une base de données, alors les informations nécessaires seront émis à un service producteur qui va générer un événement sur mesure.

Cette création va devoir respecter plusieurs règles afin de convenir à l’architecture. Tout d’abord l’événement devra contenir des étiquettes détaillants quelle(s) base(s) de données est(sont) concernée(s) par ce derniers. Cela permettra de garder actif un événement tant que tous les consommateurs concerné n’aient été atteint, de plus l’identité de l’utilisateur émetteur de l’événement sera consigné dans ce dernier pour une meilleure traçabilité et pour s’assurer que, si la modifications porte sur des données légales, la modification soit d’abord accepté par un membre de ce service.

Un fois qu’un événement est créé, il est envoyé dans le bus où il demeurera tant que toutes ses étiquettes ne soit consommées. Les consommateurs désignés par ces étiquettes vont alors recevoir l’événement et le traduire en modification qu’ils opéreront sur leurs base de données respectives.

En parallèles des intervention manuels, deux comportement automatisé surviendront :

- Le premier est un service de sauvegarde cyclique qui, toute les semaines, effectuera une sauvegarde d’un dump de la base de données légal.
- Le second est le service de centralisation et d’archivage des logs retournés par les producteurs et consommateurs d’événements.

Mise en place des changements

Dans le but de mettre cette nouvelle architecture en place tout en réduisant au maximum l'impact sur le travail des employés pendant la transition, nous allons commencé par déployé une CI/CD (Intégration continue/Livraison continue) qui nous permettra de validé les fonctionnalité via des procédures de tests avant leurs déploiement dans le système. Ensuite la nouvelle architecture reposant grandement sur le bus d'événement, il devra être déployé en second, suivi des outils de monitoring dans le but de s'assurer du bon fonctionnement des nouvelles fonctionnalités.

Une fois la nécessaire réalisé, nous pourrons commencé à développé les fonctionnalité les unes après les autres. Leurs déploiement sera progressif afin de s'assurer de la validité de chaque fonctionnalité avant de passer à la suivante pour ne pas bloqué le fonctionnement de l'entreprise.

Pour le déploiement des nouvelles fonctionnalités, nous allons nous appuyés sur deux points afin de s'assurer que leur introduction soit réalisé avec le moins d'accro possible :

- Le premier point est que l'IHM (Interface Homme Machine) sera disponible en parallèle de l'ancienne application pour chaque service. Elle ne contiendra pas toutes les fonctionnalités dans un premier temps comme décris précédemment et s'en verra agrémenter au fur et à mesure du développement. Cela permettra de familiarisé les employés progressivement à la nouvelle application.
- Le deuxième point est que nous allons installé pour la phase de transition un service RBAC (Contrôle d'Accès Basé sur le Rôle) en amont de l'IHM permettant de choisir l'attribution des fonctionnalités avant leur déploiement final à des rôles qui sont répartis parmi les employés, ce service pourra également permettre de choisir la version désiré de la fonctionnalité testée. Ce service permettra de réalisé des pré-déploiement sur une faible portion d'employés pour un test dont les retombées seront minimisé. De plus c'est également une manières d'impliquer les employés dans le développement en récupérant leurs retour d'expérience sur les nouvelles fonctionnalités. Le bonus est que si jamais un pré-déploiement se passe mal, un retour en arrière vers la dernière version stable sera simple à réaliser.

Nouvelle solution de l’architecture

Migration vers l’architecture cible

L’utilisation de la nouvelle solution sera disponible pour tous les services de l’entreprise au fur et à mesure de son développement. L’IHM sera disponible en parallèle des anciennes applications tant que les fonctionnalités de ces dernières ne seront pas totalement remplacées, comme précisé plus haut.

Une architecture de transition (figure 3 de l’annexe) sera nécessaire pour effectuer un déploiement fluide de la nouvelle solution. Elle nous permettra de déployer les fonctionnalités de la nouvelle solution au fur et à mesure, mais pour cela nous avons besoin de trois ajout temporaire absent de l’architecture d’origine.

Le premier est le RBAC (Contrôle d’Accès Basé sur le Rôle) qui sera le service nous permettant d’associer les différentes fonctionnalités à des rôles, puis d’attribuer ces rôles aux utilisateurs selon les besoins des tests de déploiement. Ainsi, nous pourrons mener des tests à petite échelle pour valider la qualité d'une fonctionnalité avant son déploiement final.

Le deuxième est le service de conversion. Il nous servira à convertir les informations reçu du consommateur au format de la base de données d’origine, si cette dernière est d’un type différent de la nouvelle.

Et enfin le troisième ajout est le service de synchronisation qui aura pour fonction de s’assurer que la nouvelle base de données reçoivent bien les changement qui pourrait être effectuer depuis l’application d’origine. Le but étant que les tests des fonctionnalités par un échantillon d’employé ne soit pas une activité séparé de leurs tâches habituelles, au contraire, ils s’agit simplement d’une nouvelle façon de les effectuer.

Avec cette architecture de transition, nous nous offrons la possibilité de fiabilisé le développement de la solution par des tests en condition réel sans perturber outre-mesure les employés des différent services.

Cette architecture de transition sera l’une des étape du processus de CI/CD et surviendra une fois que la fonctionnalité à tester aura validé les tests qui lui sont propre (test unitaires, de performance, etc.) pour valider le bon fonctionnement de cette dernière en condition réel.

Annexe :

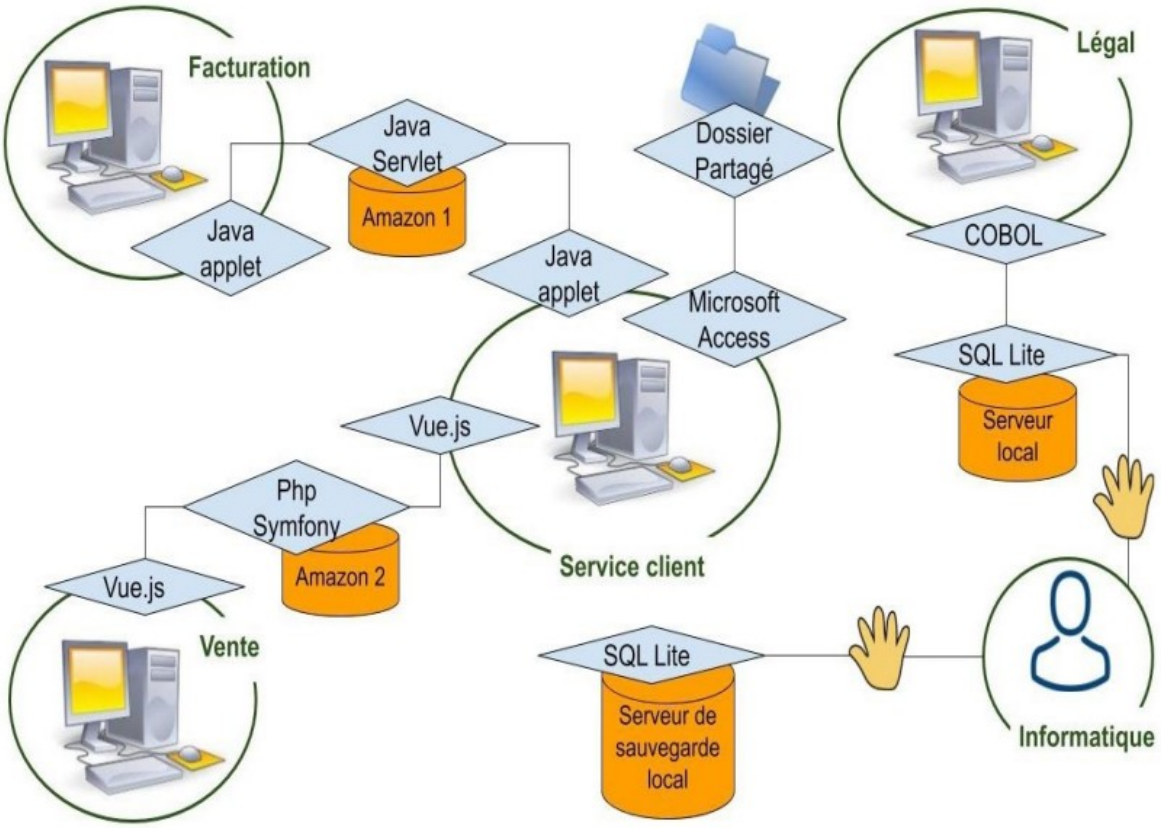

Figure 1: Architecture d'origine

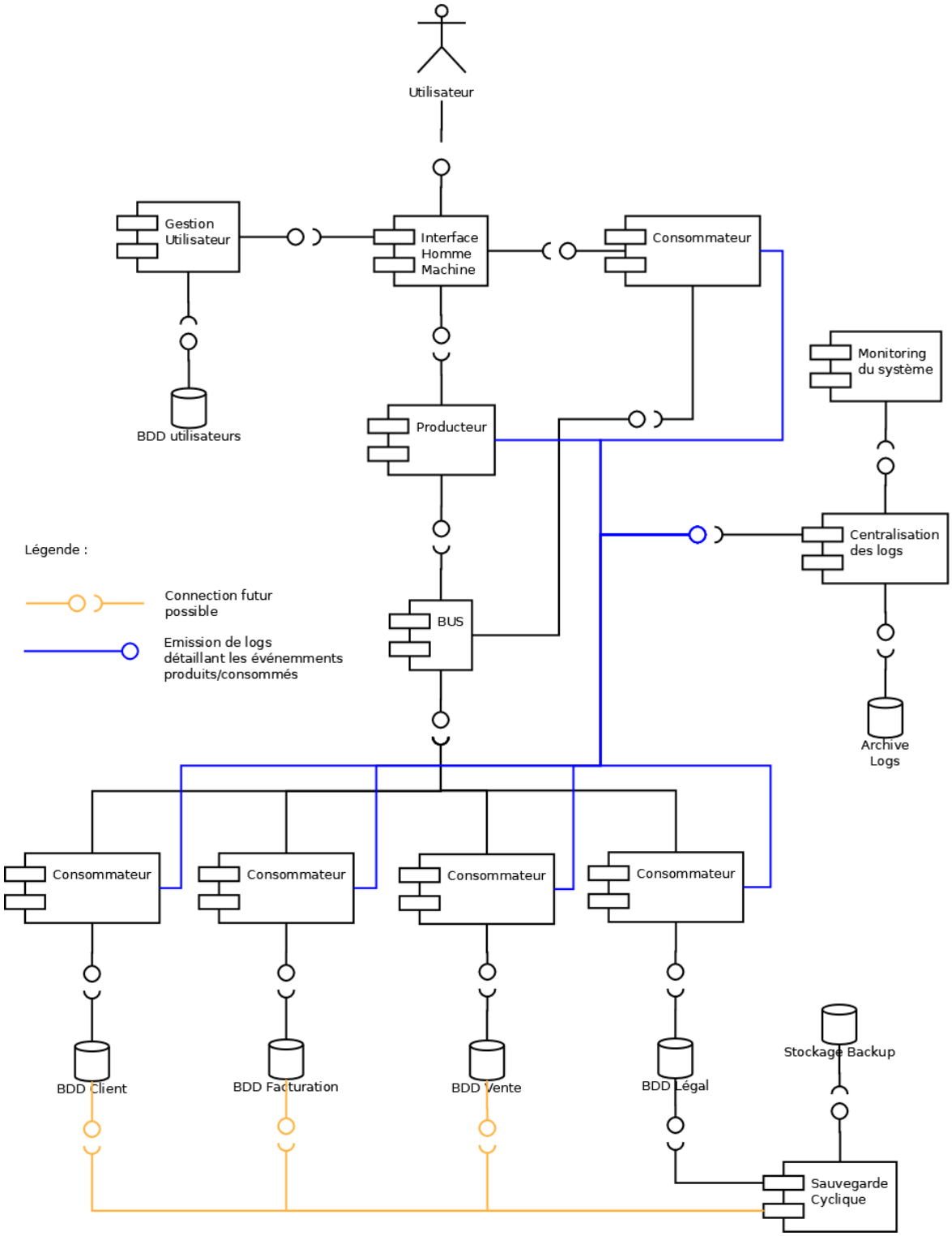

Figure 2: Architecture cible

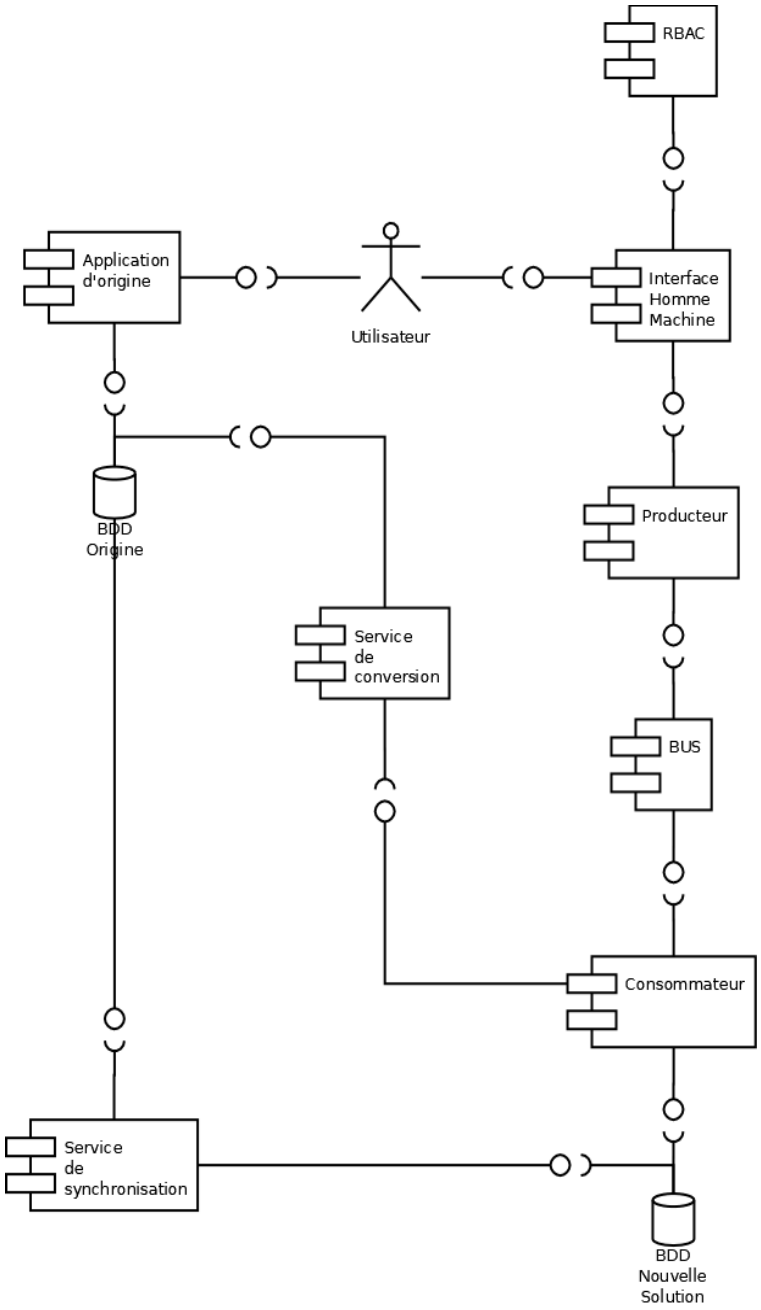

Figure 3: Architecture de transition

Préparation de l'environnement	28/01/2025	26/03/2025
Création de la CICD	28/01/2025	29/01/2025
Création du BUS	30/01/2025	12/02/2025
Création du Producteur	13/02/2025	26/02/2025
Création des consommateurs	27/02/2025	26/03/2025
Création IHM	30/01/2025	12/02/2025
Création RBAC	13/02/2025	17/02/2025
Création des BDD	18/02/2025	20/02/2025
Création Service de Conversion	21/02/2025	06/03/2025
Création du service de synchronisation	07/03/2025	18/03/2025
Développement de fonctionnalité	27/03/2025	16/07/2025
Service Client	27/03/2025	23/04/2025
Service Légal	24/04/2025	21/05/2025
Service Vente	22/05/2025	18/06/2025
Service Facturation	19/06/2025	16/07/2025

Figure 4: Détail des tâches du diagramme de Gantt, représenté dans les figure 5, 6, et 7

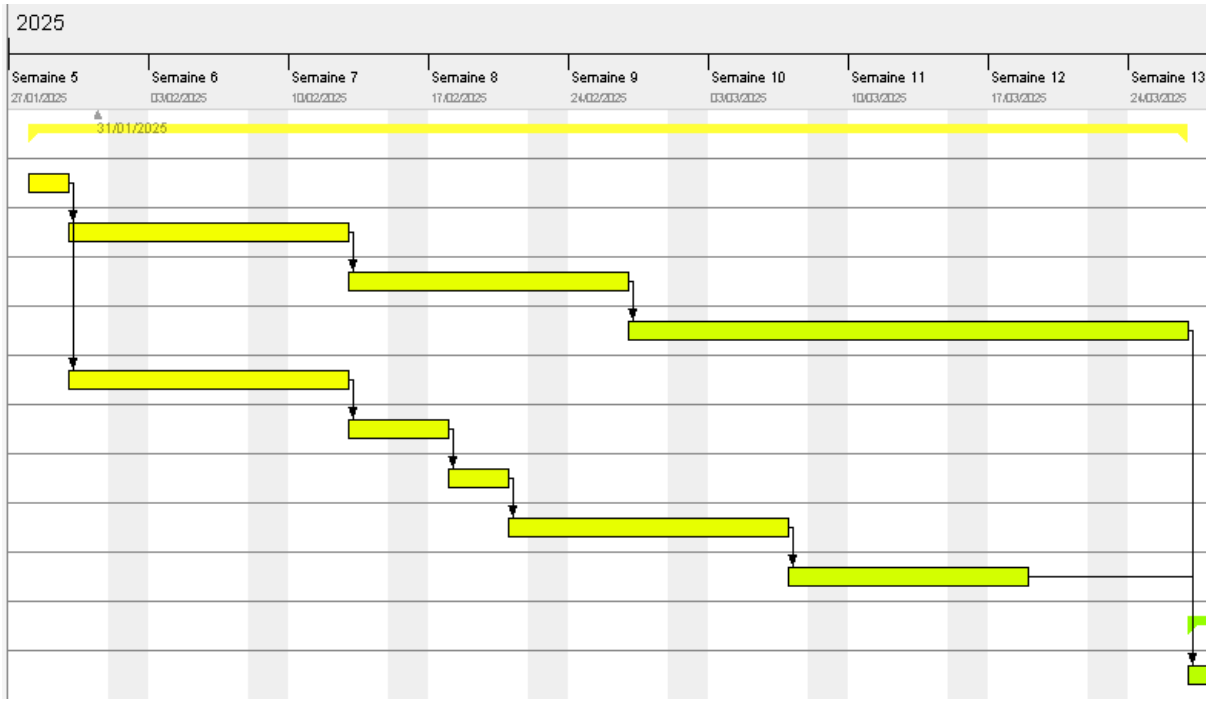

Figure 5: Première section du diagramme de Gantt représentant le phase de préparation de l'environnement

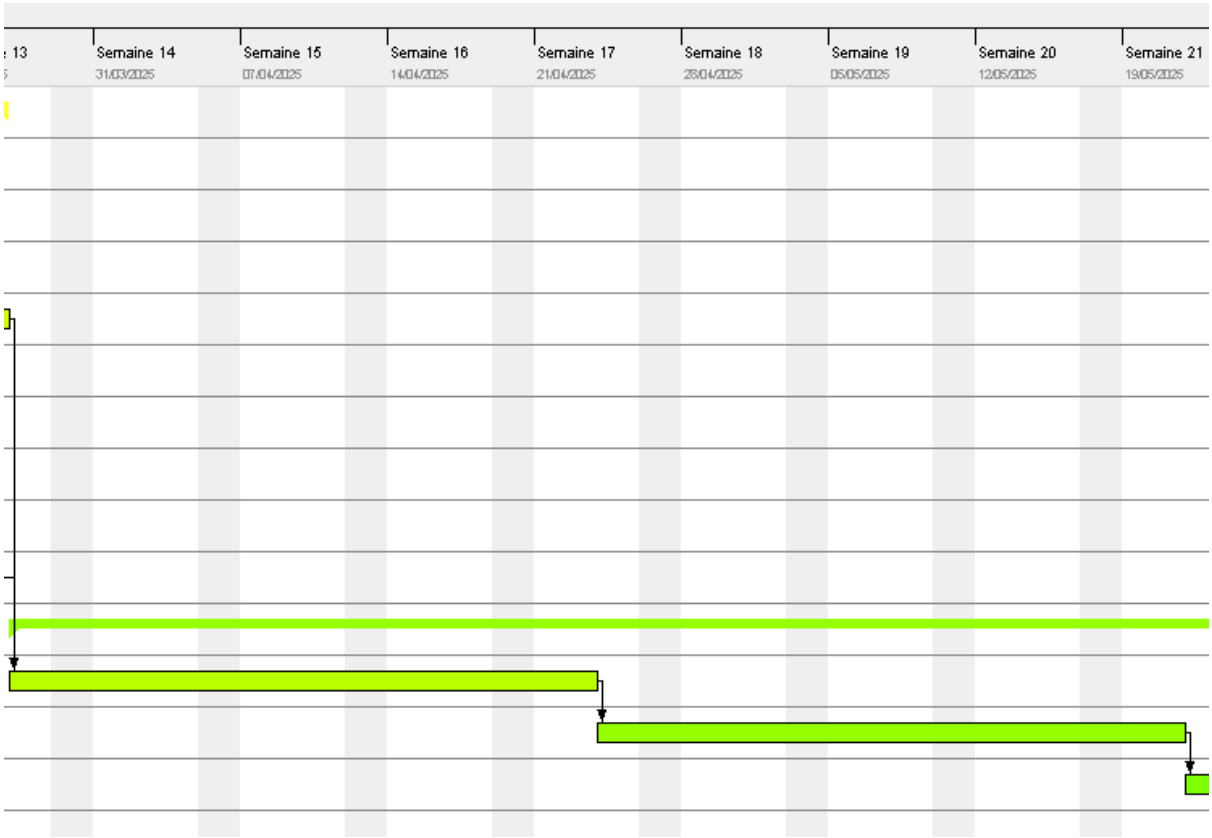

Figure 6: Deuxième section du diagramme de Gantt représentant le début de la phase de développement de fonctionnalité

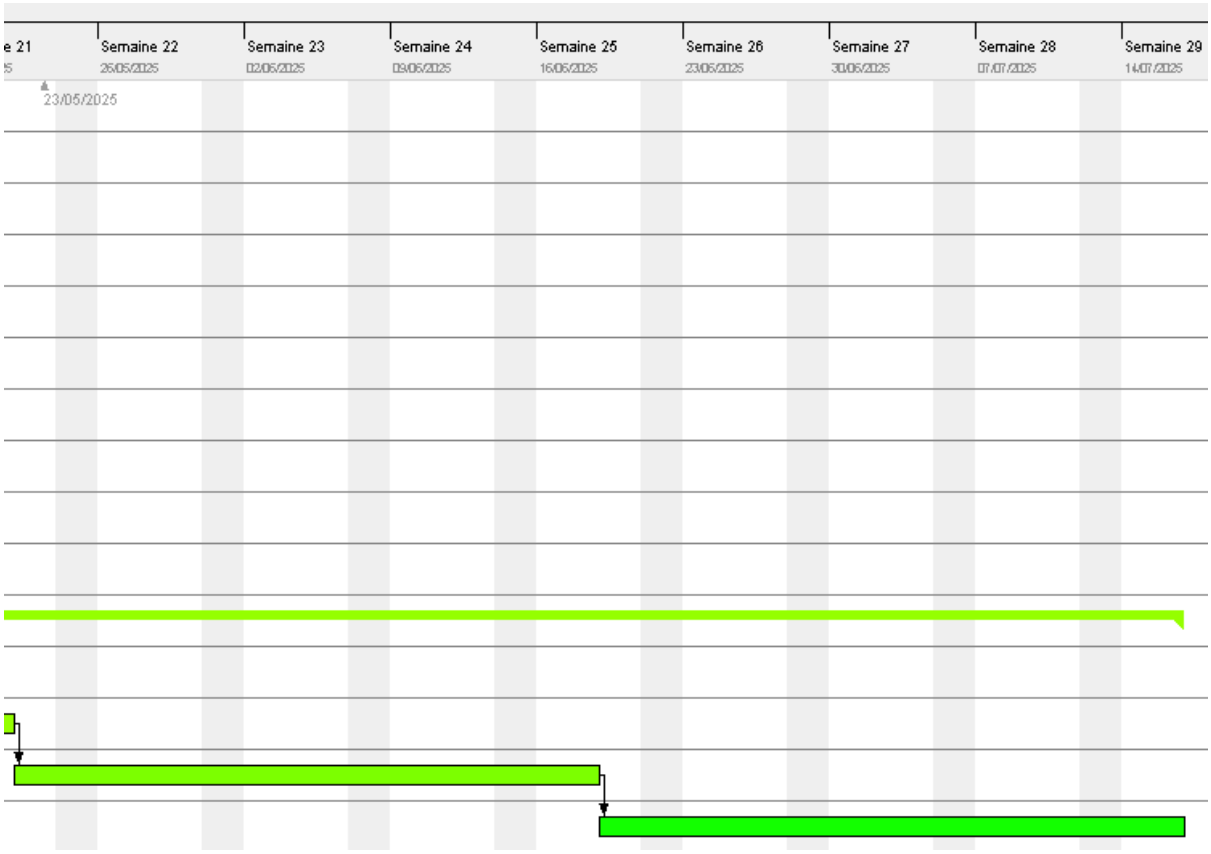

Figure 7: Troisième et dernière section du diagramme de Gantt représentant la fin de la phase de développement de fonctionnalité