**Partie 1 - « Maintien en Condition Opérationnelle puis Développement d’une Application Web en Java EE 7 »**

Contexte

Dans le cadre de ces missions, j’ai rejoint les différentes équipes en tant que développeur Java EE. Ces compétences ont été acquises lors de deux projets similaires sur les périodes d’avril 2015 à août 2017 pour le premier et de septembre 2017 à janvier 2018 pour le second.

La première expérience est le développement en pré-production puis le maintien en condition en production d’une application Web de gestion. Il s’agit d’un projet externalisé que j’ai intégré dès le début de la phase de pré-production qui a demandé six mois de délai. J’ai effectué le maintien en condition pendant deux ans et demi après sa mise en production.

La deuxième expérience est le développement d’une application Web de gestion ayant pour but d’être un bac à sable de nouvelles technologies en remplaçant un agenda collaboratif. Le projet à remplacer avait une antériorité de plus de cinq ans. Refondre ce projet avec les outils modernes permettait, en plus de se remettre à niveau sur le langage, de mettre en œuvre une méthode de développement dirigée par les tests qu’ils soient unitaires ou d’intégration.

Description de l’activité

Pour ces deux projets, j’étais en charge de corriger des bugs et d’implémenter de nouvelles fonctionnalités dans l’application dans un projet dont le squelette applicatif avait déjà été réalisé. L’équipe de maîtrise d’ouvrage était pour moi dans ces deux projets un acteur incontournable pour ce qui est de la description du besoin et la levée de doutes lorsque les spécifications fonctionnelles étaient incomplètes ou contradictoires.

La seconde expérience m’a permis de mettre pleinement ces compétences à profit et de les perfectionner en apprenant et mettant en pratique ce qui m’était alors une nouveauté : les tests unitaires et d’intégration et la qualité de code dans le processus de développement d’une application Web en Java. Les tests unitaires, avec Junit et Mockito sont rédigés avant le code afin de garantir un respect des attentes fonctionnelles, c’est la méthodologie « test-driven-development ». À ces tests unitaires nous ajoutions des tests d’intégration avec Arquillian en charge de tester le fonctionnement des méthodes avec une simulation de la réponse des appels aux méthodes externes, ces tests nécessitant alors la mise en place d’un serveur d’application embarqué. Dans la même démarche de mise à jour nous n’avions par réutilisé la solution SVN avec Tortoise-SVN pour gérer l’historisation et les versions de notre code source mais celle qui aujourd’hui est devenue un incontournable aujourd’hui, Git. Un serveur GitLab a été installé sur notre plateforme de développement interne et si nous avions commencé à travailler avec Git par l’intermédiaire de l’outil graphique git-extensions facilitant la lecture des branches, la ligne de commandes est vite devenue une habitude lorsque, bien paramétrée avec des alias par exemple, elle devient plus lisible et efficace qu’une interface graphique !

Tester le code est indispensable donc il convient d’être certain que le code soit testé dans un pourcentage minimal également nommé la couverture de code. Là aussi, ce projet a donné l’occasion de mettre en place une métrique à l’aide de SonarQube et définir un minimum de couverture de code attendu lors de la compilation est ainsi limiter la livraison de code non couvert.

Caractérisation des compétences mobilisées

La première expérience m’a permis : (i) de mettre en pratique ma formation initiale de développement, (ii) apprendre ce que sont des bonnes pratiques et les mettre en œuvre, (iii) d’évoluer vers les nouvelles itérations du langage.

Mon engouement pour ce métier a été un moteur. Je m’y suis investi personnellement, en particulier durant mon temps libre. De plus, j’ai ainsi pu participer à l’évolution des bonnes pratiques par des formations et conseils prodigués à mes collègues, essentiellement sur les nouveautés apportées par l’itération 8 du langage JAVA comme par exemple la programmation fonctionnelle (stream, interfaces fonctionnelles, consumers, prédicats etc...) l’écriture lambda ou encore l’utilisation du couple « Classe Optional / annotation Nullable » pour éliminer la bien trop connue NullPointerException.

Ces projets m’ont permis de développer l’ensemble des compétences nécessaires à la réalisation en totale autonomie d’autres projets, repoussant encore plus loin les évolutions technologies et attendus métier (cf. expérience « Partie 3 - Proof Of Concept ayant pour but de valider le choix d’une architecture logicielle »). Un résumé de ce socle minimal est l’application, le suivi et l’implication dans les règles de qualité et d’uniformisation du code, tests dirigeant le code, qualimétrie, gestion des versions, lecture et participation à l’évolution des spécifications fonctionnelles, suivi des évolutions du langage par de nombreux canaux (newsletters, revues techniques, blogs, applications mobiles, conférences physiques (POSS, Open-Source Experience) ou en visio (DEVOXX), exercices en ligne...), travail en équipe et mise en production de livrables.

**Partie 2 - « Réalisation de deux grappes de sites Internet par le CMS Joomla avec le plug-in Joomla Multi-sites puis leur intégration par script dans un serveur HTTP Apache sur une plateforme CentOS »**

Contexte

Cette expérience a débuté en août 2018 au Centre de Conduite Informatique de l’Armée de Terre. Elle a pour date de livraison le premier trimestre 2023.

J’ai d’abord été intégré à l’équipe de ce projet comme administrateur de site avant de devenir le responsable technique. L’équipe est actuellement composée d’un administrateur, de deux administrateurs-intégrateurs, d’un responsable technique, d’un chef de projet et d’une autorité cliente.

Ce projet a été lancé en DATE et a pour base un Joomla avec le plug-in Joomla Multi-sites et un template d’affichage récupéré par la précédente équipe. Ce Joomla a été installé sur un serveur Apache monté sur un serveur CentOS qui est la plateforme de développement de ce projet. A mon arrivée sur le projet, la base était finalisée, j’ai donc pu créer un site parmi les premiers livrés sur la plateforme de développement.

Description de l’activité

La création de sites demande de maîtriser l’interface d’administration du CMS Joomla mais les compétences en SQL et administration serveur demandées y sont sauf exception marginales. Réaliser un site Joomla demande énormément de tâches répétitives et la souris est donc pour cette tâche bien plus utilisée que le clavier.

Cela a changé lorsqu’il a fallu livrer sur la plateforme de pré-production les sites réalisés par ticket à l’équipe de livraison puis dépôt d’un script et des archives sur un serveur FTP. Mon équipe n’avait aucun accès à la plateforme de destination ou connaissance de sa configuration, les premiers scripts de livraison étaient donc en échec et il a fallu fournir une documentation exhaustive quels que soient les scénarii mais surtout un script shell capable de fonctionner sous CentOS ou Debian (s’agissant des deux OS serveurs autorisés par le cadre de cohérence technique fourni par la DSI de mon entreprise pour un hébergement de sites Web) et ce quel que soit son emplacement dans l’arborescence du serveur cible puisque nous n’avions pas non plus connaissance de cette information. Le travail d’intégration est donc devenu une tâche à part entière et les administrateurs en charge de créer les scripts d’intégration des sites sont donc devenus des administrateurs-intégrateurs.

Lorsque des clients ont commencé à demander une persistance de leurs modifications apportées sur leur site hébergé sur la plateforme de pré-production, les scripts ont pris en complexité puisqu’il n’était alors plus question d’effacer puis réécrire mais bien d’insérer les nouveaux sites à côté des existants, modifier et supprimer les existants le cas échéant et ce toujours sans connaître l’arborescence ou la configuration du serveur impacté. Mes bases en shell Windows m’ont permis de monter en compétence en shell Linux et ainsi réaliser ces scripts demandant de plus en plus de complexité, l’intégration étant donc sur ce projet ce qui demandait le plus de développement puisque la création de sites reste une tâche répétitive.

Un challenge nous a été imposé lorsque notre hébergeur désigné et impératif a sous-traité l’hébergement à une société qui a renégocié les contrats et demandé que nos 61 sous-sites, alors mis à jour par un seul script puisque tapant sur un seul dossier dans le serveur Apache mais ayant 61 bases de données (une par site) réclameraient donc 61 scripts différents pour chaque mise à jour. Le contrat du prix des scripts restait inchangé : 2 scripts gratuits par mois, chaque script supplémentaire étant payant (un montant à 4 chiffres par script). Nous livrions alors en moyenne, chaque mois, un script complet de livraison puis un script correctif corrigeant les inévitables erreurs ce qui nous faisait livrer gratuitement mais cela nous devenait alors impossible puisque le seul script initial, si tant est qu’il se passait bien, demanderait alors 59 fois le prix d’un script !

Nous n’avions aucune solution à part tout fusionner dans une seule base de données, les fichiers des sites étant d’ores et déjà fusionnés au sein du même dossier sur le serveur Apache.

J’ai apporté une idée dont une preuve de fonctionnement a rapidement été démontrée : la séparation des droits entre sites reviendrait à la charge de la base de données, l’utilisateur mysql d’un site disposerait alors de tous les droits sur toutes ses tables et uniquement sur ses tables qui seraient différenciées des autres tables de chaque site par un préfixe. Chaque utilisateur mysql ne disposerait donc d’aucun droit au niveau de la base de données complète sauf celui de bloquer une table pour gérer la concurrence des accès. Une fois la solution adoptée, l’appliquer sur notre plateforme de production réclame un audit de sécurité préliminaire qu’il a fallu accueillir sur des moyens qu’il nous faut configurer et mettre à disposition. Cette tâche revient normalement à l’équipe d’administrateurs système mais un turn-over récent y avait placé 2 personnes ne sachant pas effectuer cette tâche donc je m’en suis chargé.

La mise en place des moyens de l’audit préliminaire consiste à fournir un réseau fermé entre des postes clients et le serveur hébergeant nos sites. Les machines n’ayant besoin que d’un paramétrage réseau, notamment le fichier hosts.etc, tout est à faire côté serveur hébergeant les sites. Ce serveur est un LAMP sous CentOS 7 hébergé par un hyperviseur ESXi. Sur le papier, 2 semaines de configuration de cette machine virtuelle suffisent et c’est le temps maximal que j’avais pour le faire.

Si j’insiste sur cette phase c’est que la réussite de ces 15 jours était déterminante pour l’avenir du projet ! Le prochain créneau d’audit était au moins l’année suivante, une non-validation venant de la solution ou d’une incapacité à tester la solution repoussait donc d’au moins un an la mise en production de nos sites.

Le réseau étant fermé, l’absence de lien Internet aurait pu mettre en échec l’audit : j’avais passé 2 jours à récupérer à la main les très nombreuses dépendances pour CentOS 7 nécessaires à l’installation des paquets httpd, mysql-server et php puis cette procédure manuelle s’est arrêtée lorsqu’il a fallu mettre à jour des paquets critiques où toute erreur d’installation aurait pu simplement mettre le démarrage du système en échec ; heureusement un dépôt de dépendances se trouve dans l’.iso d’installation de CentOS 7 : il faut alors suivre une procédure pour monter cette .iso, déclarer le répertoire des dépendances comme dépôt par le gestionnaire de paquets de la machine virtuelle et l’installation de mes paquets et de leurs dépendances est passé sans erreur.

Une semaine vient alors de passer, je suis encore en train de configurer le serveur hébergeant les sites depuis le seul poste en ayant l’accès lorsque des jeux de malchance ont permis à quelques uns de mes savoirs de briller : ma machine a lâché ainsi que son disque dur. La machine n’a plus voulu atteindre ne serait-ce que le POST et le disque dur n’a jamais répondu même branché sur un lecteur SATA vers USB. Sur ce disque était placé la seule copie accessible de la base de données du gestionnaire de mots de passe dont ceux de l’accès SSH de la machine virtuelle hébergée par l’hyperviseur ESXi. Pour des raisons de confidentialité je n’avais pas dédié de support de stockage à une copie de cette base de données et les précédents administrateurs système affectés à la gestion de l’hyperviseur n’avaient pas transmis l’emplacement de la sauvegarde de cette archive aux nouveaux. La machine ayant lâché était la seule machine sous Windows disposant de droits administrateur de tout mon service, j’ai donc récupéré une machine Windows sous AD inutilisée et au lieu de refaire une installation complète j’ai obtenu ses accès administrateur par la création, depuis une clé usb live-CD Unix, d’une porte lançant une console de commandes au lieu du programme de gestion des touches rémanentes lors de l’écran de saisie de session où à cet endroit, les commandes exécutées disposent de privilèges élevés. J’ai ainsi pu y créer un compte, y donner son mot de passe, des droits d’administration puis m’y connecter.

La machine Windows disposant de droits administrateur obtenue, il restait encore à obtenir les accès SSH à la machine virtuelle. Cette fois, les administrateurs système disposaient bien des accès à l’interface d’administration de l’hyperviseur mais il y est impossible d’y modifier les comptes d’une machine virtuelle qui y est hébergée. Il y a là aussi un moyen d’avoir une console élevée en privilèges sous CentOS mais la méthode que je connais, inapplicable en SSH est possible via l’interface d’administration hyperviseur qui fournit la sortie console du serveur comme si j’y étais connecté de manière physique. Après avoir ralenti très fortement le démarrage de la machine virtuelle (plus de 10 secondes au POST), j’ai pu obtenir le temps nécessaire après l’ouverture de la l’interface console pour lancer mes commandes et obtenir une élévation de privilèges puis modifier le mot de passe du compte que nous utilisions en SSH. À ce moment les accès ont été rétablis et j’ai pu poursuivre la configuration.

Ceci est un résumé succinct, j’ai pu placer dans les quelques jours restants mes sites sur ce serveur et les faire répondre à temps donc l’audit a pu avoir lieu. Le jugement rendu par les techniciens évaluateurs a été favorable pour la solution adoptée qui a donc été appliquée sur les sites Internet. Aujourd’hui ce sont 61 sites Internet et 30 sites Intranet qui ont été livrés sur leur plateforme de pré-production respective et seront par la suite migrés sur une plateforme de production.

Depuis août 2021 j’occupe sur ce projet le rôle de responsable technique. Par sous-effectif cette fonction s’ajoute à ma fonction de développeur-intégrateur, fonction que je considère comme un sous-ensemble à la deuxième car nécessaire à une pleine compréhension des problématiques et enjeux techniques de la réalisation du projet. Afin de me donner du temps de travail sur cette deuxième fonction, il m’a fallu former mon équipe afin de leur transmettre toutes mes compétences utiles au projet sous peine d’être de ne pouvoir qu’être le seul de l’équipe à pouvoir travailler ce qui a été le cas sur une semaine fin 2021 en phase de livraison des sites Internet sur la plateforme de pré-production. Aujourd’hui mon équipe réalise un travail de qualité et je peux me consacrer à mon travail de manager en étant pour eux un appui technique. Cela a demandé une année entière car ma prédécesseure, en plus de n’avoir fourni en 2 ans qu’une réécriture de l’arborescence de la documentation n’a effectué aucune passation de consignes, autant de problématiques qu’il a fallu conjuguer avec les échéances de livraison attendues et la phase de formation. La gestion des priorités se fait désormais en réunion régulière avec mon chef de projet et en complément du tableau Kanban (liste des tâches à faire, en cours et faites), des « daily scrum » (« mêlée quotidienne » en traduction littérale, un rendez-vous quotidien avec toute mon équipe) ont été mis en place pour suivre l’évolution du projet, lever très rapidement des problématiques et faciliter le partage d’informations.

Caractérisation des compétences mobilisées

Les compétences demandées sur le projet ont été pour certaines abordées : administration Unix et base de données MySQL, administration d’un CMS Joomla, outils SSH tels que Putty, Vim et Nano, rédaction de scripts shell Unix, autonomie, capacité d’adaptation et d’anticipation, culture informatique, rédaction de documentation, méthodologie agile

**Partie 3 - « Réalisation en totale autonomie d’un Proof Of Concept ayant pour but de valider le choix d’une architecture logicielle en vue de réaliser un nouveau projet informatique et de garantir la viabilité de son développement par les outils fournis de l’entreprise »**

Contexte

Cette mission a été réalisée et finalisée sur la période de fin janvier 2018 à début août 2018.

Développer sur la demande client

Cela a été un succès après 6 mois de réalisation, de fin janvier 2018 à début août 2018.

Hormis le nombre de micro-services back-end, il respecte en totalité l’architecture retenue pour le projet client. Il consiste donc en 3 micro-services en Java EE 7 (JDK 8), 1 en charge du front-end et 2 en charge du back-end.

Les micro-services communiquent en REST (HTTP) par le framework Springboot et sont montés sur un serveur Wildfly, les queues asynchrones sont hébergées sur un serveur ActiveMQ, l’authentification et la sécurisation des tunnels entre micro-services est déléguée à un SSO Keycloak monté sur son propre serveur.

Le framework d’interface du front-end est Primefaces, les dépendances de tous les micro-services sont gérées par Maven et la base de données est une PostGreSQL.

Les beans Java sont gérés par injection de dépendances ce qui facilite la compréhension du code et donc le debug mais surtout la performance avec une gestion mémoire bien plus maîtrisée par un cycle de vie des objets optimal.

Toutes les bonnes pratiques de développement du moment en ma connaissance ont été appliquées : documentation complète au format JavaDoc, serialVersionUID généré pour toute classe sérialisable et incrémenté à chaque évolution de la classe pour en garantir l&apos;intégrité, contexte des beans restreint au plus élémentaire, navigation entre pages par un Flow pour limiter là-aussi la multiplication d&apos;objets et réduire au possible leur empreinte mémoire, généricité maximale…

Description de l’activité

Les contraintes du « poste informatique du développeur » sont nombreuses : Windows 10 sans aucun accès Internet ni aucun droit administrateur mais reliés avec les autres postes par un Intranet.

Un serveur regroupant des outils de développement, ce que l&apos;on appelle une forge logicielle est fournie sur cette Intradef. Elle nous donne accès à Git et SVN qui sont deux dépôts versionnés de sources, Jenkins qui, intermédiaire à Git et SVN, est un outil d’intégration continue vérifiant la qualité et le respect des attentes fonctionnelles lors d’envois de code, Nexus qui est un dépôt de dépendances, Wiki qui centralise la documentation au format « Wikipédia » et Mantis qui est un outil de suivi des anomalies et évolutions.

La problématique du non-accès à Internet vient de l&apos;accès aux dépendances résolu en partie par le dépôt de dépendances Nexus qui hors Internet est loin d’être complet. Le gestionnaire de dépendances Maven utilisé à chaque compilation identifie les éventuelles dépendances manquant à Nexus donc pour permettre la compilation elles sont rajoutées dans le repository local du poste du développeur, il faut donc les transmettre avec les sources du projet à chaque développeur intervenant sur le code. Cela n’a pas eu lieu sur ce POC mais l’a été sur le projet client.

Caractérisation des compétences mobilisées

La problématique des droits limités vient de l’installation des outils et des services. Le code créé en local sur l’IDE Eclipse a pu également être exécuté en local par un serveur Wildfly local mais le serveur d’authentification Keycloak et le serveur PostgreSQL n’ont pu être installés que sur des machines virtuelles dont la configuration après-démarrage a été automatisée par l’outil Vagrant pour faciliter le montage et remontage. Une alternative à la conteneurisation, encore interdite d’usage par la DSI de mon entreprise.

La réalisation de ce « Proof of Concept » a demandé plusieurs phases :

Description d’une architecture détaillée et interopérable

Description de la base de données

Réalisation de la base du projet avec 2 micro-services

Réalisation de la machine virtuelle de base de données

Intégration du SSO Keycloak sur sa machine virtuelle

Ajout d’un 3èmemicro-service

Ajout de traitements asynchrones