Journal d'activité

octobre 2021 janvier 2022

Projets

- Correction du tentacule Picklog de l'outil de test appelé le Kraken
- Correction de l'activité VerifyCheckSignature du tentacule C21Whitebox du kraken
- Correction de bugs graphiques sur l'interface web Parcel Tracker 5 qui permet de suivre les colis dans le système d'information

Colis 21

Colis 21 est un système d'information. Il s'agit d'un ensemble de programmes qui communiquent entre eux pour assurer une représentation informatique des réels colis en circulation afin de les tracer, les suivre et d'effectuer des actions en cas d'anomalies détectés.

Le moteur

Le moteur est le composant principal de Colis 21. Il s'agit du seul programme qui a le droit d'interagir avec la base de données directement. Tous les autres programmes doivent interagir avec lui pour pouvoir avoir des informations sur les colis. Ainsi le moteur peut traiter les évènements qu'il reçoit des autres services et prendre des décisions en fonction pour les rejeter ou les valider et collecter les nouvelles informations. Il est possible de communiquer avec le moteur en envoyant des messages dans un logiciel d'agent de message appelé RabbitMQ. Comme tous les services n'implémentent pas ce système de communication, un web service appelé ModuleWebHost permet de communiquer plus simplement via le protcole http. Le web service se charge de transformer les requêtes http en message RabbitMQ après une validation supplémentaire des données.

Le kraken

Le kraken est une interface web qui permet de tester Colis 21 en simulant des colis dans le système d'information. En réalite il s'agit d'un ensemble de

programmes utilisant le framework .Net en C#. Ces programmes sont appelés des tentacules et sont contrôlées par un programme appelé le Brain. Celui-ci est responsable de démarrer, interagir et stopper les tentacules du kraken. Chaque tentacule est fournie avec une liste d'activités. Ces activités correspondent à une simulation des étapes de la vie d'un colis et les tentacules sont en quelque sorte des catégories d'activités. Il est ensuite possible pour un testeur de combiner ces activités en définissant ce que l'on appelle un cas de test afin de représenter un scénario de la vie d'un colis et de s'assurer que le système d'information se comporte comme il devrait. Le cas de test peut ensuite être exécuté pour lancer la simulation du scénario de la vie du colis.

Parcel Tracker 5

Parcel Tracker 5 est la 5e version de l'outil Parcel Tracker. Parcel Tracker est une interface web qui permet de suivre l'évolution des colis. Celle-ci est faite avec le framework React et se base sur le web service ModuleWebHost.

Périodes

Correction de l'activité VerifyCheckSignature du tentacule C21Whitebox du Kraken

Objectifs et missions poursuivies

- Comprendre le fonctionnement du tentacule.
- Comprendre l'API de Colis21
- Comprendre le fonctionnement du moteur
- Corriger l'activité du tentacule

Actions concrètes (1 à 3 actions par objectif, 5 à 6 lignes par action)

- Reproduction du bug sur l'environnement test-v sur lequel travaille les testeurs puis en local en configurant mon environnement de façon adéquate. Avant même de me lancer dans la correction du bug j'essaie tout d'abord reproduire le bug dans l'environnement de test appelé test-v sur lequel le bug a tout d'abord été détecté. Pour cela j'ai besoin de me connecter en vpn au réseau de l'entreprise afin d'avoir accès aux différents environnements sur lesquels sont le kraken est déployé.
- Une fois le bug reproduit sur l'environnement de test j'essaie de le reproduire en local en configurant mon environnement local avec une nouvelle branche git basée sur la même branche utilisée pour l'environnement test-v et je mets à jour ma base de données mongodb par rapport à la base de données utilisée sur test-v.
- Il me faut ensuite analyser quelles sont les tentacules nécessaires à lancer pour pouvoir lancer mon cas de tests sans avoir. L'idéal est de lancer

uniquement les tentacules nécessaires au cas de test choisi et pas les autres car cela aurait pour effet de ralentir mon poste en local.

Difficultés rencontrées et solutions trouvées (environ 1 page)

Compatiblité de versions

Difficulté Pour bien configurer mon environnement de développement en local afin de corriger le bug, je dois passer par plusieurs étapes. Tout d'abord je dois aller sur le site Azure Devops qui regroupe notre code source et nos tâches à effectuer selon la méthodologie agile. Dessus je dois trouver le bug à résoudre et lui ajouter une nouvelle tâche de réalisation que je dois créer. Il faut ensuite que je crée une nouvelle branche git partant d'une branche git stable du projet qui est d'habiture master. Il faut ensuite lier cette branche à ma tâche de réalisation et récupérer ma branche en local. Une fois cela fait ce n'est pas encore suffisant pour pouvoir lancer le projet en local. En effet le Kraken est un projet qui dépend de middlewares pour fonctionner. Ces middlewares sont RabbitMQ, MongoDB et Logstash. Pour cela il faut les lancer également et grâce à une technologie de conteneurisation appelé Docker cela est rendu possible facilement depuis un fichier texte. Cependant la base de données que j'avais en local n'était pas compatible avec la branche git utilisée. Il m'a donc fallu faire une importation de la base de données depuis l'environnement de test pour m'en servir. Cela est possible grâce à une commande assez longue difficile à retenir.

Solution Comme je devais régulièrement taper la commande permettant l'importation de base de données à des versions différentes selon les branches git utilisées, j'ai réalisé un script Bash permettant de charger des alias permettant de faire cela automatiquemement avec des commandes faciles à retenir.

Échec de comparaison

Difficulté Le bug que je devais traité concernait une instance de cas de test du kraken qui avait fini en échec pour une raison inexpliquée. Pour résumer ce cas de test avait pour but de simuler la création d'un colis en envoyant une requête http spécifique que l'on appelle évènement directement au web service de colis21. Le web service reçoit la requête, vérifie sa validité et la converti en direction du moteur en message RabbitMQ. Le moteur traite ensuite le message et le stocke dans sa base de données sous forme de nouveau colis. L'activité suivante du cas de tests appelé VerifyCheckSignature avait pour but de vérifier que la signature associé au colis dans MongoDB correspondait bien à la signature fournir sur l'interface web comme entrée de l'activitée. Cependant ces signatures étant des images je me suis rendu compte d'un bug subtile. En effet cette comparaison échouait car les images comparées était différentes à 1 seul pixel près.

Solution Afin de régler ce problème j'ai sollicité l'aide d'un des Tech Leads de l'équipe pour mieux m'orienter dans la recherche de la cause du problème. En investiguant celui-ci s'est rendu que cette erreur était uniquement du à une mauvaise utilisation d'une entrée d'une des activités du cas de tes qui était mal renseigné et cela à cause d'une erreur dans la documentation des information d'entrées. Il a égalemement fallu que je définisse le format par défaut des signatures en SVG à la place du format PNG qui était utilisé auparavant.

Compétences acquises (hard et soft skills) (environ 1 page)

Hard skills J'ai pu découvrir le fonctionnement du logiciel d'agent de message RabbitMQ. J'ai pu assisté à une présentation sur l'outil qui explicait les différents avantages d'utiliser ce type de middleware. J'ai pu découvrir comment les applications peuvent créer des messages destinées à des Exchanges qui peuvent être de plusieurs types en fonction de si l'on souhaite mutiplexer les messages selon des règles prédéfinies dans des queues à destinations d'autres services consommateurs. J'ai également pu apprendre la syntaxe des requêtes pour interagir avec le serveur de base de données MongoDB.

Soft skills En passant par ces différentes étapes j'ai pu me rendre compte des différentes étapes qui permettent le déploiement et l'implémentation de nouvelles fonctionnalités dans un sytème d'information. Mais également quelles relations entretenir avec les membres de l'équipe afin d'obtenir des renseignement ou soliciter de l'aide. Cela m'a permis de renforcer la précision la description des problèmes que je rencontrait afin de permettre une résolution plus rapide et efficace.

Travail sur Parcel Tracker 5

Objectifs et missions poursuivies

- Comprendre l'organisation du code source du projet React -> Lecture du code de l'application React et expérimentation sur de nouvelles branches git pour étudier le fonctionnement des composants
- Comprendre la hiérarchie des composants React dans l'application -> Installation du plugin React-Dev-Tools sur firefox pour afficher la hiérarchie des composants dans la console de firefox
- Trouver leur définition dans le code source -> Recherche des occurrences des noms de composants en utilisant grep

Action concrètes (1 à 3 actions par objectif, 5 à 6 lignes par action)

- Déploiement de l'outil Parcel Tracker en local pour reproduire le bug en question. Pour cela je crée une branche sur Azure Devops lié à la tâche de réalisation en question. Je récupère la branche sur mon poste en local puis je mets à jour ma base de données en local en utilisant mon script de

mise à jour de base de données. Il faut ensuite lancer le middleware dont la base de données MongoDB en plus du web service et de ParcelTracker.
- J'ai du trouver les composants à modifier en installant l'extension React Dev Tools et en inspectant les éléments graphiques à corriger en les retrouvant dans la hiérarchie des composants React. Il a ensuite fallu que je fasse un inventaire des composants React que j'ai à ma disposition afin de mieux reprendre le composant qui comporte le bug. Une fois le bon composant choisir il a fallu que je retrouve la définition des composants parents afin de changer les composants enfants à utiliser et cela en fonction de la page à afficher.

Difficultés rencontrées et solutions trouvées (environ 1 page)

Analyse

Difficulté Afin de pouvoir me lancer dans la correction du bug j'avais tout d'abord besoin de bien comprendre le fonctionnement de l'application en elle-même. Comme plusieurs services faisant partie du système d'information partagent le même code source en commun, celui-ci est très fourni et il peut être assez difficile d'investiguer quels sont les zones pertinentes dans la résolution du problème. Pour cela j'ai effectué des recherches dans le block note en ligne que partage l'équipe sur One Note pour rechercher des informations concernant le projet sous un aspect technique mais aussi fonctionnel.

Solution Longues périodes d'analyse du code source et du block note contenant des informations sur la partie fonctionnelle et technique du projet

Compétences acquises (hard et soft skills) (environ 1 page)

Hard skills

- Montée en compétences sur React. J'ai pu apprendre de quelle façon les composants React sont imbriqués les uns dans les autres pour former l'affichage graphique des applications web moderne dynamique d'aujourd'hui. J'ai pu mieux comprendre comment fournir les propriétés qui permettent l'instanciation des composants et la façon dont ceux-ci sont disposés les uns dans les autres pour former des composants parents réutilisables dans plusieurs pages.

Soft skills Après avoir reçu de nombreux commentaires sur mes Pull Request je réalise mieux la façon dont travaille le service informatique des entreprises. Grâce aux différentes réunions tous les matins appelés les Daily Meeting j'ai souvent l'occasion de voir sur quels éléments travaillent les membres de mon équipe. Avec les différent sprint Scrum nous avons à chaque fin de sprint une réunion de rétrospective de sprint dans lesquels les Product Owner nous informent des

tâches qui sont prioritaires à livrer par rapport à d'autres et les assignent pour le sprint suivant lors du Sprint planning.

Glossaire

- Git: Logiciel de gestion de version d'un code source. Git permet de tracer l'évolution d'un code source grâce à un historique qui comprend une liste de commits.
- Commit: Un commit est une modification incrémentale du code source que le développeur peut nommer et appliquer à l'historique.
- Branche: Version divergente d'un code source qui comprend une nouvelle historique de commit dont les premiers sont en communs. Un projet git comprend habituellement une branche principale stable appelée "master" et chaque nouvelle fonctionnalité est développée sur une nouvelle branche git basée sur master afin d'être fusionner plus tard.
- Pull Request: Demande de fusion d'une branche git vers une autre (habituellement vers master). Une Pull Request a pour but d'être examiné par les développeurs expérimentés de de l'équipe habituellement le Responsable Technique et les Techs Lead. Si celle-ci est validée par ces membres alors les nouveaux commits ajoutés sur cette branche sont appliqués aussi sur master. On appelle cela la fusion. Remarque la fusion dans un sens et dans l'autre peuvent ne pas produire le même résultat.
- Web Service: Un web service est un serveur http avec lequel un client http peut communiquer via un standard de communication (ou interface) défini au préalable appelé API soit Application Programming Interface.
- Système d'information: Un système d'information est un ensemble de services utilisant des technologies adaptées pour interagir entre eux afin de collecter, traiter et stocker des données.
- Environnement (Informatique): Un environnement est un contexte sur une machine qui permet de déployer une copie du système d'information afin de le développer et tester ses services.
- Terminal : Application de bureau permettant d'utiliser les programmes d'une machine en tapant une commande sous forme textuelle
- Bash : Langage permettant d'exécuter des commandes pour exécuter des programmes
- React: Framework front-end permettant de rendre des applications web plus dynamiques
- Framework: Ensemble de règles et de librairies qui définissent l'organisation d'un code source pour un type d'application prédéfinit.
- Composant: Terme propre à React qui correspond à une abstraction permettant de simplifier l'organisation du code source