**MEMOIRE**

**DE STAGE DE FIN D’ETUDES**

**Pour l’obtention du**

**«Mastère professionnel en Nouvelles Technologies des Télécommunications et Réseaux (N2TR)»**

**Présenté par :**

**Hamdi FITATI**

|  |
| --- |
| **Développement d’une interface graphique, en Qt C++, pour automatiser des tests de validation** |

**Soutenu le :………………………**

**Devant le jury :**

**Président :** Mr.(Mme.)……………………………………………………………………….……..

**Encadreur :** Mr.(Mme.) ……………………………………………………………………….…..

**Rapporteur :** Mr.(Mme.)……………………………………………………………….….………

**Membre :** Mr.(Mme.)……………………………………………………………….….…………..

**Année Universitaire : 2018 / 2019**

# Résumé

En vue de l’obtention du diplôme de Mastère Professionnel en Nouvelles Technologies des Télécommunications et Réseaux « N2TR », à L’université Virtuelle de Tunis « UVT » pour l’année universitaire 2018-2019, des efforts ont été portés pour présenter ce projet qui consiste à développer une interface graphique pour automatiser des cas de test de validation et qui était réalisé au sein de la société ACTIA.

Mots clés: Validation, Automatisation, CAN, Embarquée, Qt C++, Ranorex

**Abstract**

In order to obtain the diploma of Professional Master in New Technologies of Telecommunications and Networks "N2TR", at the Virtual University of Tunis "UVT" for the academic year 2018-2019, efforts have been made to present this project which consists of developing a graphical interface to automate validation test cases and which was realized at ARDIA

Key words: Validation, Automation, CAN, Embedded, Qt C ++, Ranorex

**ملخص**

# Dédicaces

# Remerciement

Il est particulièrement agréable, avant de présenter ce travail, d’exprimer notre gratitude envers les personnes qui, de près ou de loin nous ont apporté leur aide inestimable lors de la réalisation de ce projet de fin d’études. Nous tenons à témoigner toute notre gratitude et notre haute considération à :

Monsieur **Najdi BEN HASSINE**, mon encadrant au sein de l’entreprise, pour sa disponibilité, son aide, ses conseils précieux, ses critiques constructives, ses explications et suggestions pertinentes ainsi que pour ses qualités humaines et morales que j’ai toujours appréciées et respectées.

Madame **Ahlem BEN YOUNES**, enseignante à l’université virtuelle Tunis UVT, pour m’avoir permis de bénéficier de son grand savoir dans la matière, pour sa pédagogie, ses compétences, et son aide précieux tout au long de ce projet.

Toutes les personnes qui ont contribué à l’élaboration de ce travail.

Notre dernier mot s’adresse aux membres du jury, Monsieur xxxxx et Monsieur

xxxxxx, pour l’honneur qu’ils nous font de bien vouloir accepter d’évaluer ce travail.

# Table des matières

[Résumé I](#_Toc11443103)

[Dédicaces II](#_Toc11443104)

[Remerciement III](#_Toc11443105)

[Table des matières IV](#_Toc11443106)

[Liste des figures VII](#_Toc11443107)

[Liste des tableaux VIII](#_Toc11443108)

[Liste des Acronyms IX](#_Toc11443109)

[Introduction générale 1](#_Toc11443110)

[Chapitre 1 : Contexte du projet 2](#_Toc11443111)

[Introduction 3](#_Toc11443112)

[1. Cadre général 3](#_Toc11443113)

[2. Organisme d’accueil 3](#_Toc11443114)

[2.1. Présentation du GROUPE ACTIA : 3](#_Toc11443115)

[2.2. Présentation d’organisme d’accueil ARDIA : 4](#_Toc11443116)

[3. Etude de l’existant 6](#_Toc11443117)

[3.1. Description du projet TGU 6](#_Toc11443118)

[3.2. Critique de l’existant 6](#_Toc11443119)

[3.3. Solution proposée 7](#_Toc11443120)

[4. Choix méthodologique 7](#_Toc11443121)

[4.1. Méthodologie SCRUM 8](#_Toc11443122)

[4.2. Les rôles dans une équipe SCRUM 8](#_Toc11443123)

[4.3. Les réunions SCRUM 9](#_Toc11443124)

[4.4. Formalisme adopté 9](#_Toc11443125)

[Conclusion 10](#_Toc11443126)

[Chapitre 2 : Etude préliminaire du projet 11](#_Toc11443127)

[Introduction 12](#_Toc11443128)

[1. Spécification des besoins 12](#_Toc11443129)

[1.1. Module de gestion des paramètres 12](#_Toc11443130)

[1.2. Module de connexion 12](#_Toc11443131)

[1.3. Module de gestion des trames CAN 12](#_Toc11443132)

[1.4. Module de gestion des fichiers spéciaux 12](#_Toc11443133)

[1.5. Module de génération des logs 13](#_Toc11443134)

[2. Identification des acteurs 13](#_Toc11443135)

[3. Diagramme de cas d’utilisation global 13](#_Toc11443136)

[4. Spécifications non fonctionnelles 14](#_Toc11443137)

[5. Planification et Back log du produit 15](#_Toc11443138)

[Conclusion 15](#_Toc11443139)

[Chapitre 3 : Conception du projet et environnement de développement 16](#_Toc11443140)

[Introduction 17](#_Toc11443141)

[1. Conception architecturale du système 17](#_Toc11443142)

[1.1. Architecture logique 17](#_Toc11443143)

[1.2. Architecture technique 18](#_Toc11443144)

[2. Environnement de développement 19](#_Toc11443145)

[2.1. Environnement matériel 19](#_Toc11443146)

[2.2. Environnement logiciel 19](#_Toc11443147)

[2.3. Framework et technologies 20](#_Toc11443148)

[Conclusion 22](#_Toc11443149)

[Chapitre 4 : Conception détaillée du projet – Release 1 23](#_Toc11443150)

[Introduction 24](#_Toc11443151)

[1. Sprint 1 : Gestion de la connexion 24](#_Toc11443152)

[1.1. Back log 24](#_Toc11443153)

[1.2. Analyse 24](#_Toc11443154)

[1.3. Conception 26](#_Toc11443155)

[1.4. Réalisation 26](#_Toc11443156)

[2. Sprint 2 : Gestion des paramètres 28](#_Toc11443157)

[2.1. Back log 28](#_Toc11443158)

[2.2. Analyse 28](#_Toc11443159)

[2.3. Conception 33](#_Toc11443160)

[2.4. Réalisation 34](#_Toc11443161)

[3. Sprint 3 : Gestion des trames 38](#_Toc11443162)

[3.1. Back log 38](#_Toc11443163)

[3.2. Analyse 38](#_Toc11443164)

[3.3. Conception 42](#_Toc11443165)

[3.4. Réalisation 43](#_Toc11443166)

[Conclusion 47](#_Toc11443167)

[Chapitre 5 : Conception détaillée du projet – Release 2 48](#_Toc11443168)

[Introduction 49](#_Toc11443169)

[1. Sprint 4 : Gestion des fichiers 49](#_Toc11443170)

[1.1. Back log 49](#_Toc11443171)

[1.2. Analyse 49](#_Toc11443172)

[1.3. Conception 52](#_Toc11443173)

[1.4. Réalisation 53](#_Toc11443174)

[2. Sprint 5 : Génération des logs 55](#_Toc11443175)

[2.1. Back log 55](#_Toc11443176)

[2.2. Analyse 55](#_Toc11443177)

[2.3. Conception 57](#_Toc11443178)

[2.4. Réalisation 57](#_Toc11443179)

[3. Sprint 6 : Automatisation d’un cas de test 57](#_Toc11443180)

[3.1. Cas de test : 58](#_Toc11443181)

[3.2. Script Ranorex : 58](#_Toc11443182)

[Conclusion 59](#_Toc11443183)

[Conclusion générale 60](#_Toc11443184)

[Bibliographie 61](#_Toc11443185)

[Annexe 62](#_Toc11443186)

# Liste des figures

[**Figure 1: Logo de la société ACTIA** 3](#_Toc11442939)

[**Figure 2: Logo de la société ARDIA** 4](#_Toc11442940)

[**Figure 3: l’implantation du groupe ACTIA/ACTIELEC Technologies** 4](#_Toc11442941)

[**Figure 4: Siège social de la société ARDIA** 6](#_Toc11442942)

[**Figure 5: Architecture** **Embarquée** 6](#_Toc11442943)

[**Figure 6: Explication de la méthodologie SCRUM** 8](#_Toc11442944)

[**Figure 7: Diagramme des Acteurs** 13](#_Toc11442945)

[**Figure 8: Diagramme de cas d'utilisation global** 14](#_Toc11442946)

[**Figure 9: Architecture logicielle** 17](#_Toc11442947)

[**Figure 10: Architecture du modèle MVC** 18](#_Toc11442948)

[**Figure 11:Icône du Qt Creator** 19](#_Toc11442949)

[**Figure 12: Icône du Ranorex Studio** 20](#_Toc11442950)

[**Figure 13: Icône du C++** 21](#_Toc11442951)

[**Figure 14: Icône de C#** 21](#_Toc11442952)

[**Figure 15: Environnement de développement** 22](#_Toc11442953)

[**Figure 16: Gestion de la connexion** 25](#_Toc11442954)

[**Figure 17: Diagramme de séquence de "Se Connecter"** 26](#_Toc11442955)

[**Figure 18: Interface de connexion** 26](#_Toc11442956)

[**Figure 19: Connexion - Cas de la réussite** 27](#_Toc11442957)

[**Figure 20: Connexion - Cas de la failure** 27](#_Toc11442958)

[**Figure 21: Gestion des paramètres** 28](#_Toc11442959)

[**Figure 22: Diagramme de séquence de « Réinitialiser, lire, écriture et sauvegarder la modification d’un paramètre"** 33](#_Toc11442960)

[**Figure 23: Diagramme d'activité de "Ajouter un paramètre"** 34](#_Toc11442961)

[**Figure 24: Affichage de tous les paramètres** 35](#_Toc11442962)

[**Figure 25: Affichage du contenu d'un paramètre (En mode lecture et écriture)** 35](#_Toc11442963)

[**Figure 26: Affichage du contenu d'un paramètre (En mode lecture seule)** 36](#_Toc11442964)

[**Figure 27: Affichage du contenu d'un paramètre booléen** 36](#_Toc11442965)

[**Figure 28: Modification de la valeur actuelle d'un paramètre** 37](#_Toc11442966)

[**Figure 29: Interface pour contrôler l'accès** 37](#_Toc11442967)

[**Figure 30: Interface pour ajouter ou modifier un paramètre** 38](#_Toc11442968)

[**Figure 31: Gestion des trames** 39](#_Toc11442969)

[**Figure 32:Diagramme d'activité du sprint "Gestion des trames**" 43](#_Toc11442970)

[**Figure 33: Remplissage manuel des données** 44](#_Toc11442971)

[**Figure 34: Remplissage automatique des données** 44](#_Toc11442972)

[**Figure 35: Remplissage des données avec BAM** 45](#_Toc11442973)

[**Figure 36: Envoyer une seule trame une seule fois** 46](#_Toc11442974)

[**Figure 37: Envoyer une seule trame plusieurs fois** 46](#_Toc11442975)

[**Figure 38: Envoyer plusieurs trames** 47](#_Toc11442976)

[**Figure 39: Gestion des fichiers** 49](#_Toc11442977)

[**Figure 40: Diagramme de séquence du "Importer et Supprimer les fichiers"** 52](#_Toc11442978)

[**Figure 41: Diagramme d'activité du sprint "Gestion des fichiers"** 52](#_Toc11442979)

[**Figure 42: Enregistrer les fichiers** 53](#_Toc11442980)

[**Figure 43: Attacher un fichier** 53](#_Toc11442981)

[**Figure 44: Fichier traduit** 54](#_Toc11442982)

[**Figure 45: Elément trouvé** 54](#_Toc11442983)

[**Figure 46: Elément non trouvé** 55](#_Toc11442984)

[**Figure 47: Génération des logs** 56](#_Toc11442985)

[**Figure 48: Diagramme de séquence du "Génération des logs"** 57](#_Toc11442986)

[**Figure 49: Exemple de log généré** 57](#_Toc11442987)

[**Figure 50: Automatisation des tests** 58](#_Toc11442988)

# Liste des tableaux

[**Tableau 1: Planning des itérations** 15](#_Toc11442989)

[**Tableau 2: Back log du Sprint 1** 24](#_Toc11442990)

[**Tableau 3: Description textuelle de "Se Connecter"** 25](#_Toc11442991)

[**Tableau 4: Back log du sprint 2** 28](#_Toc11442992)

[**Tableau 5: Description textuelle de "Lire la valeur d'un paramètre"** 29](#_Toc11442993)

[**Tableau 6: Description textuelle de "Ecrire une valeur dans un paramètre"** 29](#_Toc11442994)

[**Tableau 7: Description textuelle de "Sauvegarder les modifications"** 30](#_Toc11442995)

[**Tableau 8: Description textuelle de "Réinitialiser les paramètres"** 30](#_Toc11442996)

[**Tableau 9: Description textuelle de "Ajouter un paramètre"** 31](#_Toc11442997)

[**Tableau 10: Description textuelle de "Modifier un paramètre"** 32](#_Toc11442998)

[**Tableau 11: Description textuelle de "Supprimer un paramètre"** 32](#_Toc11442999)

[**Tableau 12: Back log du sprint 3** 38](#_Toc11443000)

[**Tableau 13: Description textuelle de "Envoyer une trame une seule fois"** 40](#_Toc11443001)

[**Tableau 14: Description textuelle de "Envoyer une trame plusieurs fois"** 41](#_Toc11443002)

[**Tableau 15: Description textuelle de "Envoyer un ensemble de trame"** 41](#_Toc11443003)

[**Tableau 16: Description textuelle de "Envoyer une trame avec le protocole BAM"** 42](#_Toc11443004)

[**Tableau 17: Back log du sprint 4** 49](#_Toc11443005)

[**Tableau 18: Description textuelle du "Imprter les fichiers"** 50](#_Toc11443006)

[**Tableau 19: Description textuelle du "Supprimer les fichiers"** 50](#_Toc11443007)

[**Tableau 20: Description textuelle du "Traduire le contenu du fichier"** 51](#_Toc11443008)

[**Tableau 21: Description textuelle du "Rechercher un élément dans le fichier "** 51](#_Toc11443009)

[**Tableau 22: Back log du sprint 5** 55](#_Toc11443010)

[**Tableau 23: Description textuelle du "Générer les logs"** 56](#_Toc11443011)

[**Tableau 24: Cas des tests possibles pour l'exigence "Excès de vitesse"** 58](#_Toc11443012)

# Liste des Acronyms

**ARDIA** : ACTIA Group Recherche et Développement en Informatique Appliquée

**UVT** : Université Virtuelle de Tunis

**N2TR** : Nouvelle Technologie et Télécommunication Réseaux

**QA**: Quality Assurance

**TGU**: Telematic Gateway Unit

**IQCP** : Ingénierie, qualification et certification produit

**ROI**: Return On Investment

**IHM**: Interface Homme Machine

**UML**: Unified Modeling Language

**FMS**: Fleet Management System

**CAN**: Controller Area Network

**SAE**: Society of Automotive Engineers

# Introduction générale

Un logiciel n’est pas seulement du code, mais aussi des besoins, une gestion de projet, une spécification, une conception, un code source, un exécutable et des tests ! Le but de ces tests est de vérifier les exigences de la spécification logicielle de haut niveau et la satisfaction du client. C’est dans ce cadre que j’ai réalisé mon projet de fin d’études au sein de la société ARDIA.

L’intitulé du mon projet est « Développement d’une interface graphique en QT C++ qui permet d’automatiser des tests de validation ». Comme son nom l’indique, je suis sensé de rendre assister les tests logiciels c’est-à-dire au lieu de lancer des tests manuels, nous développons des scripts des tests à exécuter automatiquement.

L’étape de test a souvent été le parent pauvre du processus de production des logiciels.

Mais avec l’arrivée de systèmes de plus en plus complexes et les exigences clients, les tests sont devenus primordiaux. C’est dans ce contexte que l’entreprise ARDIA a décidé d’utiliser des outils de test au sein de l’équipe validation Software.

Le rapport est organisé comme suit : Le premier chapitre intitulé « Contexte du projet » porte sur la présentation de l’organisme d’accueil, étude de l’existant et la méthodologie de travail adoptée. Le deuxième intitulé « Etude préliminaire du projet », est dédié à l’analyse et la modélisation des besoins fonctionnels et non fonctionnels de notre projet. Le troisième chapitre « Conception du projet et environnement de développement » présente l’architecture, l’environnement matériel et logiciel ainsi que les différentes technologies utilisées lors du développement. Le quatrième chapitre « Conception détaillée » détaille la conception de la solution proposée ainsi que la réalisation de chaque sprint. Finalement, une conclusion générale qui résume et met en valeur les apports de cette expérience professionnelle et illustre de possibles perspectifs.

# Chapitre 1 : Contexte du projet

*« Le projet est le brouillon de l'avenir.*

*Parfois, il faut à l'avenir des centaines de brouillons. »*

*Jules Renard, écrivain*

# Introduction

Au début de ce rapport, il s’avère évident d’exposer le contexte général de notre projet de fin d’étude. Ainsi, nous présentons l’organisme d’accueil qui est la société ARDIA sise à la Zone Industrielle Chotrana, en l’introduisant et mettant la lumière sur ses différents axes d’activités, son organigramme et ses différentes missions et services. Ensuite, nous dégageons la problématique liée à notre projet pour aboutir aux objectifs fixés par l’entreprise. Par la suite, nous présenterons la méthodologie adoptée pour effectuer ce travail.

# Cadre général

Le présent projet intitulé « Développement d’une interface graphique en QT C++ qui permet d’automatiser des tests de validation», a été réalisé dans le cadre de la préparation du projet de fin d’études en vue de l’obtention du diplôme de Master Professionnel en Nouvelle Technologie et Télécommunication Réseaux « N2TR » au sein de l’Université Virtuelle de Tunis « UVT». Ce projet a été réalisé dans la société ARDIA.

# Organisme d’accueil

## Présentation du GROUPE ACTIA :

Le groupe ACTIA fut fondé, à Toulouse en 1986, par certains cadres de la société BENDIX. A l’origine, les principales activités du groupe ACTIA étaient la conception et la fabrication des cartes électroniques en petites séries au profit des industries. Sa naissance coïncide avec l’avènement de l’électronique dans le domaine automobile. Suite cela, les dirigeants décidaient d’investir dans ce nouveau créneau, à savoir : le diagnostic automobile, l’électronique embarquée pour véhicules industriels et le diagnostic électronique automobile constituent, jusqu’à nos jours, les principaux axes stratégiques du groupe ACTIA. Afin de renforcer sa capacité de production et de maîtriser la qualité de fabrication des produits conçus jusqu’à lors par le Bureau d’Etudes de Toulouse, ACTIA acquiert, en 1988, le groupe ALCYON (composé de deux usines). [1]

Au début des années 90, ACTIA opte pour une diversification dans le domaine de la Télécommunication en rachetant la société SODIELEC.

La figure 1 présente le logo de la société ACTIA.



**Figure 1: Logo de la société ACTIA**

L’union des deux sociétés était fructueuse et a donné naissance à ACTIELEC. Fusionnant à son tour avec la société MORS, ACTIELEC devient ACTIELEC Technologies en 2000. Afin de soutenir la croissance du groupe dans les activités informatiques et électroniques automobiles, les dirigeants d’ACTIA prennent la décision de créer, en 2005, la société ARDIA à Tunis.

La figure 2 présente le logo de la société ARDIA. [1]



**Figure 2: Logo de la société ARDIA**

Avec 21 sociétés implantées dans 15 pays partout dans le monde, et des représentants dans plus de 140 pays, la dimension internationale du groupe ACTIA est une force. ACTIA s’est construite autour d’une culture internationale orientée clients. L’objectif premier des sociétés du groupe est de proposer aux clients internationaux un support local tout en permettant l’accès à l’ensemble des compétences, expertises et ressources du groupe. [1]

La figure 3 présentant l’implantation du groupe ACTIA/ACTIELEC Technologies dans le monde.



**Figure 3: l’implantation du groupe ACTIA/ACTIELEC Technologies**

## Présentation d’organisme d’accueil ARDIA :

ARDIA acronyme d’ACTIA Group Recherche et Développement en Informatique Appliquée, est une société de services et de conseil en ingénierie implantée au Technopôle ELGhazala. Elle constitue une filiale très importante localisée dans le territoire Nord-Africain. [2]

Depuis sa création, ARDIA accompagne ses clients dans le développement et l’intégration de l’électronique embarquée communicante et des logiciels ou progiciels liés ce qui lui a permis de développer un savoir-faire notable dans divers domaines.

ARDIA est constituée de plusieurs départements. On peut les classer comme suit :

* Département IQCP (Ingénierie, qualification et certification produit) c’est un laboratoire de qualification dont le rôle est de vérifier la conformité des produits aux normes et réglementations internationales.
* Département développements matériels : constitué par le service conception mécanique, le support industriel, le support manufacturier, et le test fonctionnel.
* Département Systèmes Embarqués : propose des services de Conception, et de développement des logiciels embarqués sur des différents plateformes hardware.
* Département Développements logiciels : S’occupe de la conception, la réalisation et la maintenance des logiciels spécifiques.
* Département Diagnostic Automobile : développe les outils de diagnostic automobile pour des grandes marques automobiles.
* Département Validation : C’est un département comportant deux services : validation embarquée et Software qui appliquent les techniques de vérification et de validation pour les logiciels et les matériels développés.

Notre stage de fin d’études a été réalisé au sein du service « Validation Software » du département « Validation ».

La figure 4 représente le siège social de la société ARDIA au parc technologique El-

Ghazala.



**Figure 4: Siège social de la société ARDIA**

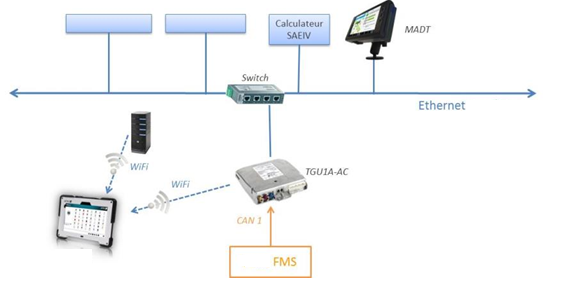
# Etude de l’existant

Mon projet se situe dans le cadre d’automatisation des tests de validation du projet Telematic Gateway Unit « TGU »

## Description du projet TGU

Le projet « TGU » consiste à valider un calculateur embarqué destiné pour les grands véhicules BUS et Camion. Il a comme objectif d’envoyer des informations concernant le véhicule et/ou le conducteur à un serveur distant pour faciliter la gestion des flottes automobiles. Comme illustré dans la figure ci-dessous, la récupération de ces informations se fait à partir des trames CAN et des autres modules communiquant avec le véhicule via l’Ethernet tel que le compteur des passagers, les calculateurs SAE.

Le périmètre de validation chez ARDIA est de tester manuellement la partie CAN qui consiste à simuler des trames du standard FMS, tel que la distance parcourue, l’Oïl et la vitesse moteur, et voir le comportement de la carte embarquée envers ces trames CAN.



**Figure 5: Architecture** **Embarquée**

## Critique de l’existant

C’est vrai que les tests fonctionnels manuels sont nécessaires, et le seront toujours, en informatique comme dans les autres domaines. Imaginez une voiture qui serait vendue aux clients sans aucun essai sur route avec un conducteur humain.

Néanmoins, beaucoup de tests relativement basiques doivent être réalisés très régulièrement durant le cycle de vie d'un logiciel, ce qui rend leur exécution manuelle fastidieuse pour un ROI (Return On Investment) faible. La durée moyenne de l’exécution d’un test manuel est de 16 minutes. Ci-dessous, un exemple des étapes pour exécuter un test manuel :

1. Ouvrir la console Putty
2. Entrer l’adresse IP du calculateur et le port pour établir la connexion
3. Entrer le login
4. Entrer le mot de passe
5. Changer la valeur du paramètre EXCSPEED à 90 Km/h
6. Changer la valeur du paramètre EXCSPDFILTER à 6000 ms
7. Sauvegarder la modification
8. Ouvrir le logiciel PeakCAN View
9. Créer la trame de la vitesse avec une valeur de 91 Km/h
10. Envoyer la trame pour 6 secondes
11. Revenir à la console Putty
12. Afficher les messages générés
13. Vérifier qu’un message relatif à l’excès de vitesse est généré avec les données correspondantes.

En plus de ce temps d’exécution qui semble assez important, l’automatisation des tests pour le projet « TGU » nécessite un logiciel qui coûte très cher. Nous citons par exemple le logiciel CANoé que sa licence coûte environ 11000 € ! D’où vient le besoin de notre solution.

## Solution proposée

En tant qu’un bon connaisseur du projet TGU, nous avons proposé à la société ARDIA une interface graphique, développée en Qt C++, qui sert à automatiser tous les plans de validation et ceci à travers :

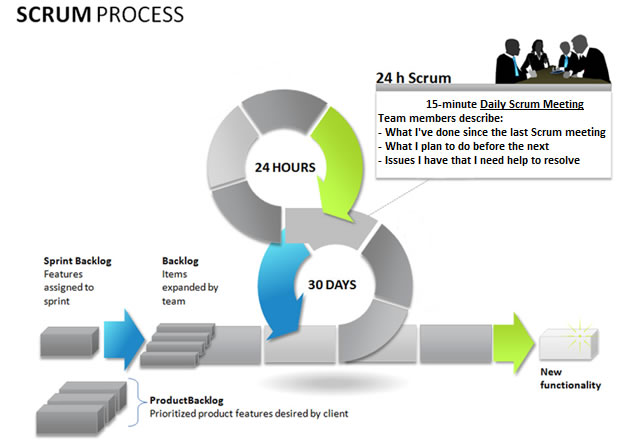
* Paramétrer le calculateur Embarquée : La gestion des différents paramètres (Lecture, Ecriture, Ajout, Modification et Suppression)
* Envoyer des trames CAN : La gestion des différentes trames CAN du standard FMS (PGN, SPN, BAM et TimeOut)
* Manipuler les fichiers spéciaux (.eco) : la reformulation des fichiers générés par le TGU pour les rendre lisibles à tout le monde.

# Choix méthodologique

Le choix d’une méthodologie agile est le choix le plus répandu pour les différents types de projets. La méthode agile se base sur un cycle de développement qui porte sur l’interaction avec le client, donc il est impliqué dans la réalisation du début jusqu’à la fin du projet. L’implication du client dans le processus permet à l’équipe d’obtenir un feedback régulier afin d’appliquer directement les changements nécessaires.

## Méthodologie SCRUM

La méthode de travail adoptée tout au long du projet est la méthodologie SCRUM, qui se caractérise par des cycles rapides de développement et de tests. Cette méthodologie de programmation consiste à travailler toujours en équipe, d'évaluer ce qui est fait d'une manière hebdomadaire et de réaliser des versions livrables chaque périodes (Release). Cette méthode augmente la productivité et minimise le temps de développement. La figure ci-dessous explique le processus à respecter en travaillant avec cette méthodologie.



**Figure 6: Explication de la méthodologie SCRUM**

Le cycle de vie SCRUM est rythmé par des itérations de quelques semaines, les Sprints. Une réunion quotidienne est faite pour consulter l'avancement de tous les membres de l'équipe ce qui permet une synchronisation totale. [3]

## Les rôles dans une équipe SCRUM

La méthode SCRUM définit trois rôles dans un projet.

* **Le « Product Owner »** : il s’agit du représentant officiel du client au sein d’un projet SCRUM. Il est l’interlocuteur principal du SCRUM Master et des membres de l’équipe. Il définit les besoins du produit et rédige les spécifications. Il est également chargé de définir et prioriser les « User Stories » pour chaque sprint.
* **Le SCRUM Master** : c’est une personne chargée de veiller à la mise en application de la méthodologie et au respect de ses objectifs. Il ne s’agit pas d’un chef de projet, mais d’une personne chargée de lever les obstacles éventuels qui empêcheront l’avancement de l’équipe et du projet pendant les sprints.
* **L’équipe** : ce sont les personnes chargées de la réalisation du sprint et d’un produit utilisable en fin de sprint. Il peut s’agir de développeurs, architectes, personnes chargées de faire des tests fonctionnels...
* Dans notre projet, le Product Owner est l’encadrant de la société alors que le Scrum Master et l’équipe sont représentés par nous-même.

## Les réunions SCRUM

Au cours de chaque Sprint, nous étions amenés à assister à des réunions régulières :

* **Réunion quotidienne** : Le « Daily Meeting » se déroule chaque matin d’une durée de quinze minutes, où le SCRUM Master anime cette réunion en affichant le Back log. Durant cette réunion, chacun à tour de rôle, mentionne ce qu’il a fait hier, tout en identifiant les problèmes rencontrés, et qu’est-ce qu’il compte faire aujourd’hui.
* Dans notre cas, et vu l’indisponibilité des différentes parties nous avons choisi de faire des réunions hebdomadaires.
* **Réunion de planification du Sprint** : Le « Sprint planning » se déroule pendant deux heures à la fin de chaque Sprint, c’est-à-dire chaque deux ou trois semaines, vendredi, animée par le SCRUM Master. Durant cette réunion, le Product Owner nous présente les sujets à traiter durant le sprint qui commence le lundi suivant, l’équipe vote, à tour de table, pour juger la complexité et l’estimation en jours pour chaque tâche.
* **Réunion de rétrospective du Sprint** : Cette réunion se déroule une demi-heure au début de chaque Sprint (n+1), chacun mentionne les éléments accélérateurs et perturbateurs rencontrés tout au long du sprint. De plus, une démonstration sera présentée pour l’équipe pour suivre l’avancement du projet.

## Formalisme adopté

Pour la réalisation de notre projet, nous avons utilisé le Langage de Modélisation Unifié « UML » qui est un langage de modélisation graphique à base de pictogrammes conçu pour fournir une méthode normalisée pour visualiser la conception d'un système. Il est couramment utilisé en développement logiciel et en conception orientée objet.

# Conclusion

Tout au long de ce chapitre, nous avons présenté l’organisme d’accueil ARDIA, ses services et son organigramme. Par ailleurs, nous avons pu dégager le contexte général du projet et présenter le choix de la méthodologie de développement. Le chapitre suivant sera consacré à la phase de planification mettant l’accent sur l’étude préliminaire des besoins fonctionnels et techniques ainsi que la présentation du Back log et la planification du projet.

# Chapitre 2 : Etude préliminaire du projet

*« Le nouveau ne sort pas de l’ancien, mais apparaît*

*à côté de l’ancien, lui fait concurrence jusqu’à le ruiner. »*

*Joseph Schumpetert*

# Introduction

Les fonctionnalités implémentées par tout système doivent correspondre à un ensemble de besoins définis par le client. Ces besoins sont les services que le client s’attend à voir fournis par le système.

À ce sujet nous définirons dans ce chapitre les besoins fonctionnels. Ensuite, d’après cette spécification, nous identifions les acteurs et nous démontrons de suite le diagramme de cas d’utilisation général de notre application. Puis, nous détaillons les besoins techniques. Le chapitre s’achèvera par une présentation du Back log du produit.

# Spécification des besoins

Afin d’assurer une meilleure compréhension des besoins et la réalisation des prototypes des interfaces, nous avons eu recours aux diagrammes de cas d’utilisation d’UML 2.5. Ces derniers permettent de spécifier formellement les besoins requis par l’application.

Dans cette section, nous analysons les différentes fonctionnalités du système. Nous les détaillons de façon informelle selon les modules suivants :

## Module de gestion des paramètres

Ce module consiste à manipuler les paramètres de la carte embarquée afin de configurer cette dernière selon les cas souhaités. Cette manipulation se traduit par la lecture, l’écriture, l’ajout, la modification et la suppression des paramètres.

## Module de connexion

Il existe deux types de calculateur dans le projet TGU, c’est les calculateurs TGU1A et TGUR. Ce module gère la méthode d’accès à ces deux types en utilisant le protocole Telnet, au cas d’un TGU1A, et le protocole SSH, au cas d’un TGUR.

## Module de gestion des trames CAN

L’utilisateur de notre interface graphique peut choisir un ou plusieurs trames CAN à envoyer grâce à ce module qui permet de gérer les formats des trames, leurs cycles d’envoi, leurs timeouts, leurs signaux et le protocole BAM.

## Module de gestion des fichiers spéciaux

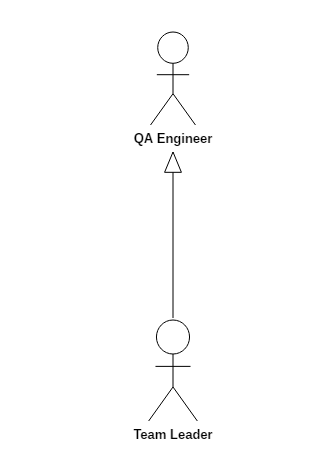
Selon les données des trames CAN reçues, le calculateur embarqué génère un ou des fichiers spéciaux contenant certaines informations concernant le véhicule, le conducteur, la maintenance, etc. Ce module permet de reformuler ces fichiers pour une lisibilité meilleure et compréhensible.

## Module de génération des logs

Notre application permet, grâce à ce module, de générer en temps réel et à la fin de chaque utilisation un fichier log qui contient l’ensemble des actions faites durant la dernière utilisation.

# Identification des acteurs

Un acteur est une entité physique (personne) ou abstraite (logiciel) capable d’utiliser l’application dans le but de répondre à un besoin bien défini. Pour notre cas, nous avons pu identifier, avec notre Product Owner, deux utilisateurs potentiels de l’application :



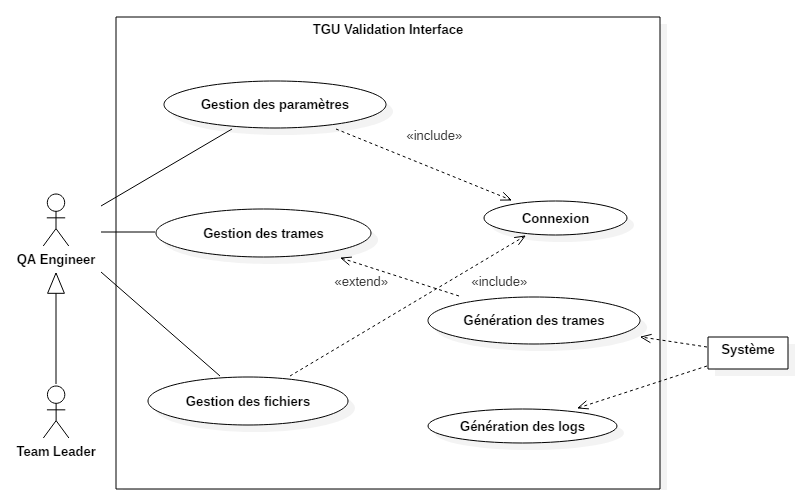
**Figure 7: Diagramme des Acteurs**

* **Ingénieur QA** : C’est la personne qui valide le projet TGU au sein de la société ARDIA
* **Team Leader** : C’est la personne qui a la main, outre que l’utilisation normale de l’application, d’ajouter, de modifier ou de supprimer un paramètre.

+ Système : Permet d’effectuer des tâches automatiques pour répondre aux différents besoins tels que l’envoi des trames CAN, la génération des fichiers logs, …

# Diagramme de cas d’utilisation global

Dans cette section, nous allons structurer les fonctionnalités de base du système par un diagramme de cas d’utilisation global, illustré par la figure 8, qui permet de donner une vision globale du comportement fonctionnel de notre système.



**Figure 8: Diagramme de cas d'utilisation global**

# Spécifications non fonctionnelles

Les besoins non fonctionnels sont des besoins qui ont un aspect visible pour l’utilisateur, et exigés par l’application. Parmi ces besoins techniques, nous pouvons citer :

* **L’Ergonomie** : C’est un élément important dans notre application, nos interfaces doivent être conviviales, simples et organisées avec cohérence en respectant les règles de l’IHM, de façon à garantir une prise en main rapide des fonctionnalités et une aisance d’utilisation.
* **Performance** : L’application doit être performante et efficace en minimisant le temps de réponse et de chargement. Il faut également réduire le temps de traitement et les délais des importations et exportations des données.
* **Modularité** : L’application doit être structurée en modules pour assurer une meilleure lisibilité, un état évolutif et extensible, une diminution du risque d’erreur et une possibilité de tests sélectifs.
* **Portabilité** : L’application doit être portable sur tous les systèmes d’exploitation (Windows, Linux et Mac)

# Planification et Back log du produit

Le Back log du produit est l’artefact le plus important de SCRUM, c’est l’ensemble des caractéristiques fonctionnelles et techniques qui constituent le produit souhaité. Les caractéristiques fonctionnelles sont appelées des histoires utilisateur (user story) et les caractéristiques techniques sont appelées des histoires techniques (technical story).

Pour le traitement de nos user story nous choisissons de commencer avec les cas d’utilisation les plus prioritaires et ayant le risque le moins élevé.

Les cas d’utilisation ou « User stories » possèdent chacun un effort qui est l’estimation initiale sur la quantité de travail nécessaire pour implémenter cette exigence. Cet effort est calculé en point d’histoire. En effet, deux points de complexité correspondent à une journée de développement.

Pour faire cette estimation d’effort nous passons par des réunions de planification des itérations qui sont les événements les plus importants dans la planification des sprints Scrum. Le but de ces réunions est de préparer le planning de travail et d’identifier le back log des itérations.

Le tableau suivant présente le calendrier de réalisation qui montre les différents sprints de notre projet :

|  |  |  |
| --- | --- | --- |
| **Sprints** | **Priorité** | **Contenu de la livraison** |
| Sprint 1 | Haute | Gestion de la connexion |
| Sprint 2 | Haute | Gestion des paramètres |
| Sprint 3 | Haute | Gestion des trames |
| Sprint 4 | Haute | Gestion des fichiers |
| Sprint 5 | Moyenne | Génération des logs |
| Sprint 6 | Haute | Automatisation d’un cas de test |

**Tableau 1: Planning des itérations**

# Conclusion

Dans ce chapitre, nous avons décrit le système comme étant une boîte noire. Nous avons étudié les principales fonctionnalités que notre application doit accomplir, nous avons encore identifié les acteurs du système puis nous avons présenté le diagramme de cas d’utilisation global et enfin nous avons fourni le Back log du produit. Nous entamons le chapitre suivant par la conception générale et nous présentons l’environnement de développement.

# Chapitre 3 : Conception du projet et environnement de développement

*« La science des projets consiste à prévenir les difficultés de l’exécution »*

*Vauvenargues*

# Introduction

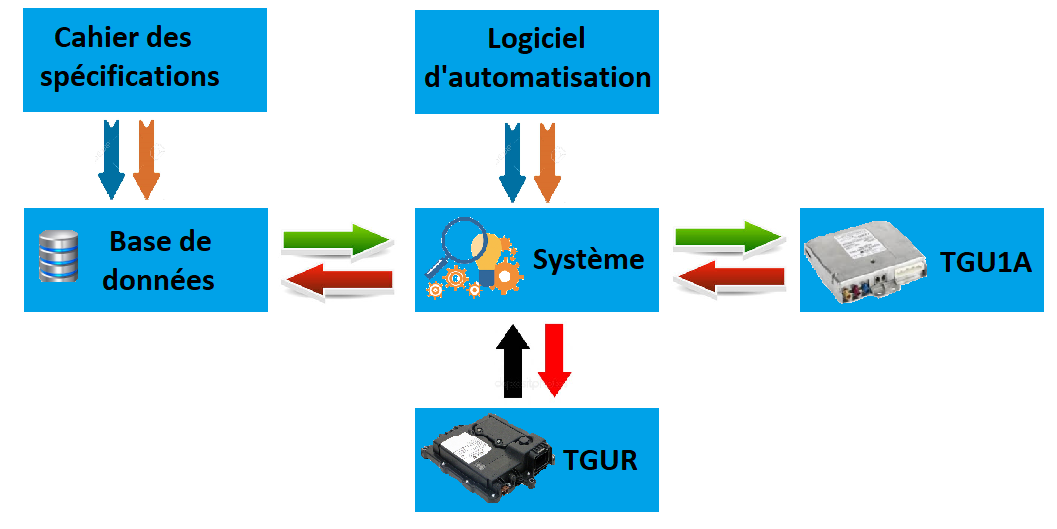
De nombreux systèmes fonctionnent dans un environnement multi-tiers, parmi lesquels se situe notre système. Ainsi, ce chapitre sera dédié, dans un premier lieu, à la conception qui est une phase très importante dans le cycle de développement d’un système et qui décrit son architecture logique et technique, et dans un second lieu, nous allons nous préparer à la réalisation en citant les différentes technologies utilisées.

# Conception architecturale du système

Il est primordial à la conception de tout système informatique de choisir le modèle d’architecture qui lui est adéquat et pouvant assurer un bon fonctionnement et une meilleure performance. De plus, l’architecture doit garantir la réutilisation et favoriser l’interconnexion fiable de ce système avec d’autres.

## Architecture logique

Le système proposé repose sur une architecture n-tiers. C’est un modèle logique d’architecture applicative qui vise à séparer nettement n couches logicielles au sein d’un même système. La figure ci-dessous représente l’interconnexion entre notre application et les autres systèmes pour réaliser ce qui est attendu.

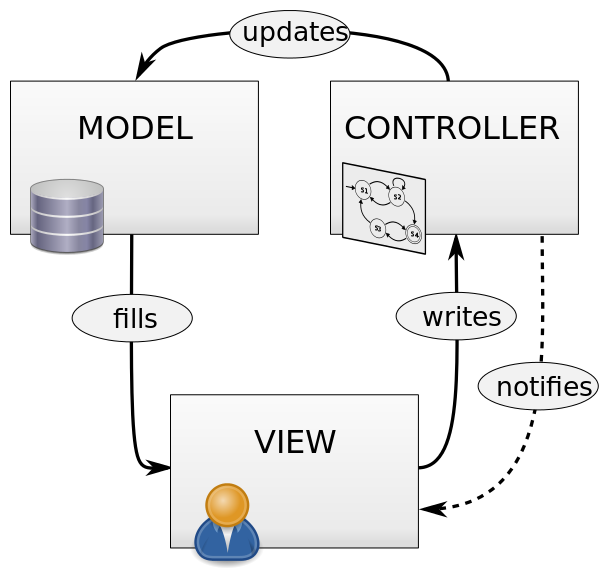


**Figure 9: Architecture logicielle**

* **Cahier des spécifications** : C’est le point d’entrée de tous les projets de validation chez la société ARDIA. Il alimente, dès le démarrage de la phase de développement de notre application, la base de données.
* **Base de données** : Il s’agit d’un fichier Excel qui contient toutes les données nécessaires pour le fonctionnement de notre application. Elle sera mise à jour à chaque évolution de la spécification à partir de notre application.
* **Application** : C’est notre système à développer.
* **Cartes embarquées** : Ce sont les calculateurs sous tests. La communication entre notre système et ces unités sera établie par les deux protocoles SSH et TELNET.
* **Logiciel d’automatisation** : C’est le logiciel qui assurera l’automatisation des tests de validation en utilisant notre application.

## Architecture technique

MVC est le modèle de conception le plus populaire surtout lorsqu’il s’agit d’une meilleure décomposition des composants d’une application.



**Figure 10: Architecture du modèle MVC**

La figure 10 ci-dessus représente l’architecture technique adoptée pour ce système et se compose de trois parties :

* **Le modèle (Model)** : qui représente les données de l’application. Il définit aussi l’interaction avec la base de données et le traitement de ces données.
* **La vue (View)** : qui représente l’interface utilisateur. Elle n’effectue aucun traitement, elle se contente simplement d’afficher les données que lui sont fournies par le modèle.
* **Le contrôleur (Controller)** : qui gère l’interface entre le modèle et le client. Il va interpréter la requête de ce dernier pour lui envoyer la vue correspondante. De plus, il effectue la synchronisation entre le modèle et les vues.

# Environnement de développement

Nous présentons dans ce qui suit l’environnement matériel et logiciel ainsi que le framework utilisé dans notre projet.

## Environnement matériel

La société ARDIA a mis à notre disposition pour la réalisation du projet un ordinateur de bureau dont la configuration est la suivante :

* ***Processeur***: i5-7200U CPU Intel Core up to 2.7 GHz
* ***Mémoire RAM*** : 8 Go
* ***Disque Dur*** : 1 To

## Environnement logiciel

Durant la réalisation de notre projet, nous avons utilisé les logiciels suivants :

* **Qt Creator**: C’est un environnement de développement intégré multiplate-forme faisant partie du framework Qt. Il est donc orienté pour la programmation en C++. Il intègre directement dans l'interface un débogueur, un outil de création d'interfaces graphiques, des outils pour la publication de code sur Git et Mercurial ainsi que la documentation Qt. L'éditeur de texte intégré permet l'auto complétion ainsi que la coloration syntaxique. Il utilise sous Linux le compilateur gcc. Qt Creator peut utiliser MinGW ou le compilateur de Visual Studio sous Windows [4]



**Figure 11:Icône du Qt Creator**

* **Ranorex Studio** : C’est un outil puissant pour l’automatisation des tests. Il s’agit d’un cadre d’automatisation de test d’interface graphique utilisé pour tester des applications Web, de bureau et mobiles. Il n’a pas son propre langage de Scripting pour automatiser les applications. Il utilise des langages de programmation standard tels que VB.Net et C#. Il peut automatiser n’importe quel ordinateur de bureau, application Web, ou logiciel mobile [5]



**Figure 12: Icône du Ranorex Studio**

## Framework et technologies

* **FrameWork :**

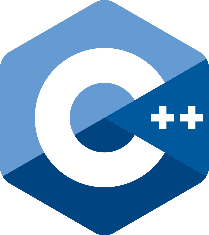
**Qt** : C’est un framework initialement prévu pour faciliter la création d'interfaces graphiques pour le langage de programmation C++. Au fil du temps et des nouveaux apports aux bibliothèques de Qt, ce framework s'est étendu progressivement au-delà des interfaces graphiques pour fournir une bibliothèque de composants facilitant l'utilisation du langage C++ (sockets, fichiers, structure de données, threads, synchronisation, ...), allant jusqu'à permettre la portabilité du code source à différentes plateformes. La portabilité des applications n'utilisant que des composants Qt se fait par simple recompilation du code source. Les environnements supportés sont les Unix (dont Linux) qui utilisent le système graphique X Window System, Windows et Mac OS X [6]

**Protocole de communication TELNET** : C’est un protocole utilisé sur tout réseau TCP/IP, permettant de communiquer avec un serveur distant en échangeant des lignes de texte et en recevant des réponses également sous forme de texte

**Protocole de communication SSH :** C’est à la fois un programme informatique et un protocole de communication sécurisé. Le protocole de connexion impose un échange de clés de chiffrement en début de connexion. Par la suite, tous les segments TCP sont authentifiés et chiffrés. Il devient donc impossible d'utiliser un sniffer pour voir ce que fait l'utilisateur

* **Technologies :**

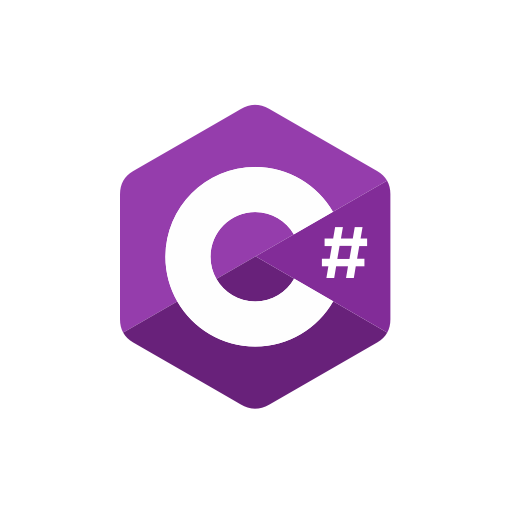
**C++ :** C’est un langage de programmation compilé permettant la programmation sous de multiples paradigmes (comme la programmation procédurale, orientée objet ou générique). Ses bonnes performances, et sa compatibilité avec le C en font un des langages de programmation les plus utilisés dans les applications où la performance est critique [7]



**Figure 13: Icône du C++**

**C# :** C’est un langage de programmation orientée objet, commercialisé par Microsoft depuis 20022 et destiné à développer sur la plateforme Microsoft .NET.

Il est dérivé du C++ et très proche du Java dont il reprend la syntaxe générale ainsi que les concepts, y ajoutant des notions telles que la surcharge des opérateurs, les indexeurs et les délégués. Il est utilisé notamment pour développer des applications web sur la plateforme ASP.NET [8]



**Figure 14: Icône de C#**

La figure suivante représente la liaison entre les différentes technologies utilisées.



**Figure 15: Environnement de développement**

# Conclusion

Nous avons présenté dans ce chapitre l’architecture logique et technique de la solution à réaliser, ainsi que le Framework et les technologies qui ont mis en place les bases de développement de notre application. Dans le chapitre suivant, nous précisons la liste des tâches à exécuter dans chaque sprint, en passant par l’analyse, la conception détaillée et la réalisation.

# Chapitre 4 : Conception détaillée du projet – Release 1

*« Chaque bonne réalisation, grande ou petite,*

*connait ses périodes de corvées et de triomphes; un début,*

*un combat et une victoire. »*

*Mahatma Gandhi*

# Introduction

Ce chapitre représente les détails des sprints réalisés tout au long de notre projet de fin d’études. Le projet s’étale sur six sprints. Les trois premiers vont être décrits dans ce chapitre par leurs back logs et expliqués durant toutes les phases d’analyse, de conception et de réalisation. A la fin de chaque sprint, nous aurons un produit stable représenté par des captures d’écran.

# Sprint 1 : Gestion de la connexion

Ce premier sprint consiste à établir la connexion entre notre interface et les deux types de cartes embarquées.

## Back log

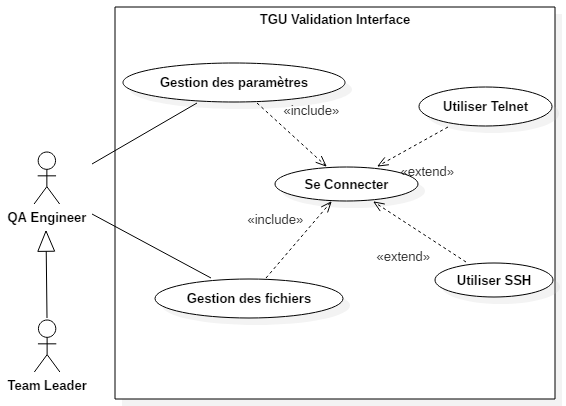
Le tableau suivant représente la planification du premier Sprint.

|  |  |
| --- | --- |
| **USER Story** | **Tâches** |
| Se connecter | Se connecter en Telnet |
| Se connecter en SSH |

**Tableau 2: Back log du Sprint 1**

## Analyse

Le diagramme de cas d’utilisation, représenté par la figure suivante, met en valeur les grandes lignes de la connexion.



**Figure 16: Gestion de la connexion**

Dans ce sprint, le QA Engineer doit avoir la possibilité de se connecter aux cartes embarquées.

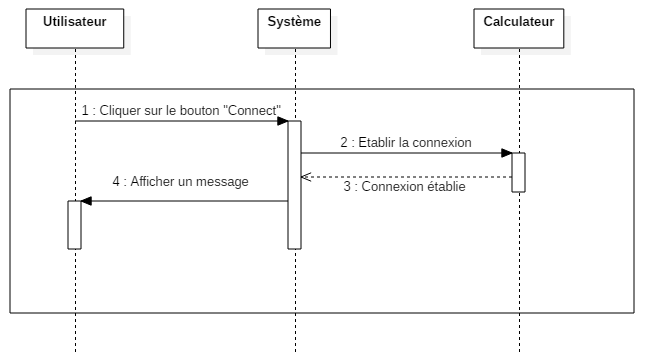
Le tableau ci-dessous présente une description textuelle du cas d’utilisation « Se connecter ».

|  |  |
| --- | --- |
| Acteur | Utilisateur (QA Engineer ou Team Leader) |
| Résumé | Se connecter à un calculateur en utilisant les protocoles Telnet ou SSH selon le type de la carte. |
| Précondition | Le calculateur est lié à l’ordinateur via un câble Ethernet |
| Post-condition | Le calculateur est connecté à notre système |
| Scénario nominal | 1. L’utilisateur choisit le type du calculateur 2. L’utilisateur clique sur le bouton « Connect » 3. Le système établie une connexion selon le type du calculateur choisi 4. Le système affiche un message dans la barre d’état disant que la connexion est établie |
| Exceptions | Connexion échouée |

**Tableau 3: Description textuelle de "Se Connecter"**

## Conception

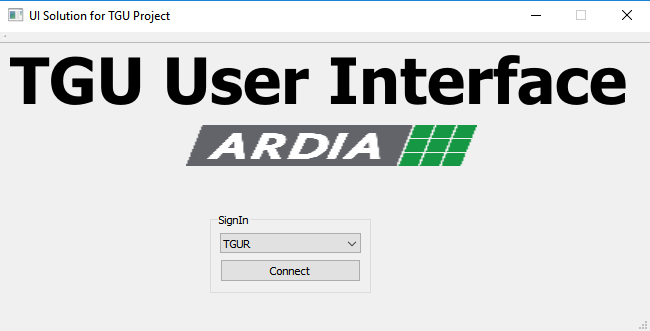
Le cas d’utilisation de « Se Connecter » est détaillé par la figure 16 suivante qui représente le déroulement de ce scénario.



**Figure 17: Diagramme de séquence de "Se Connecter"**

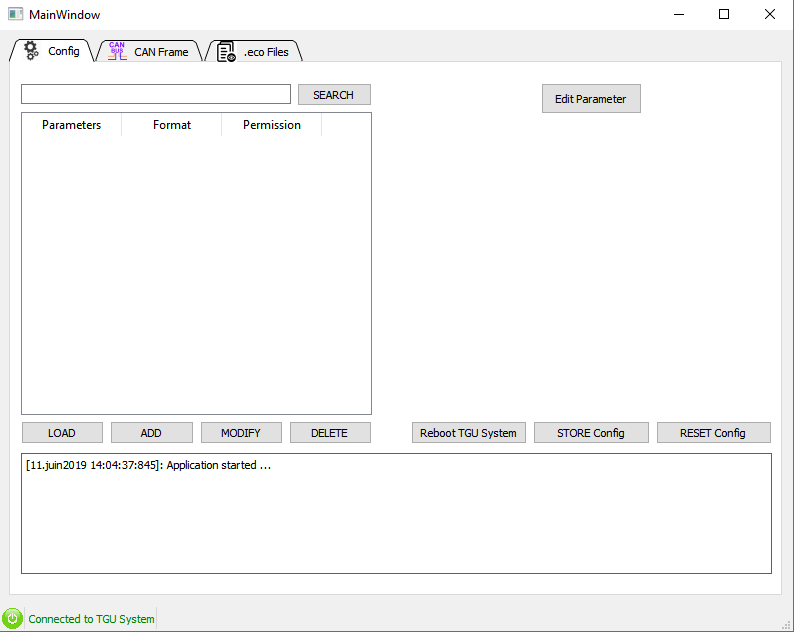
## Réalisation

A la fin de ce sprint, nous avons pu présenter à notre Product Owner une application qui permet d’établir une connexion Telnet ou SSH selon le type de la carte utilisée. Comme illustré dans la figure suivante, l’utilisateur doit choisir à partir d’une liste déroulante le type de la carte à utiliser, soit TGUR ou TGU1A.

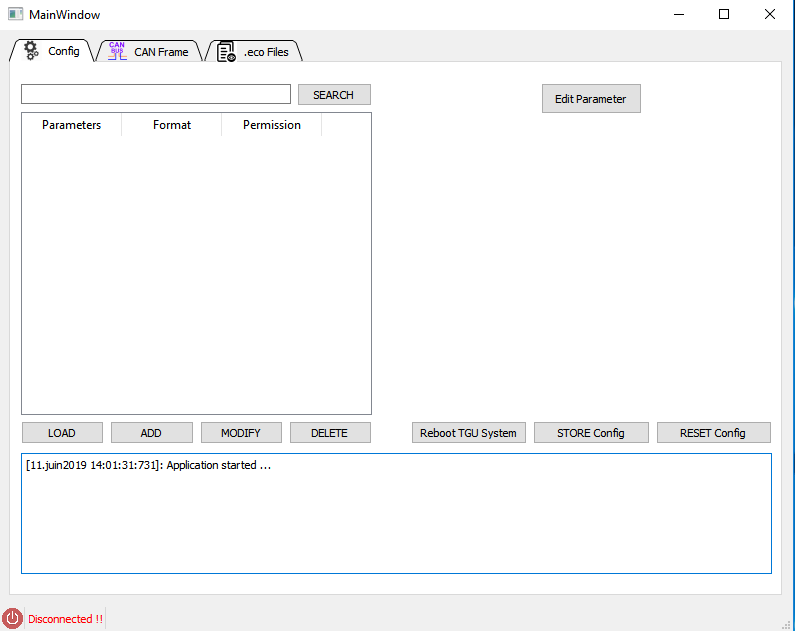


**Figure 18: Interface de connexion**

Après avoir cliqué sur le bouton « Connect », une deuxième interface s’affiche en indiquant, dans sa barre d’état, le statut de la connexion. Ceci est représenté par les deux figures ci-dessous.



**Figure 19: Connexion - Cas de la réussite**



**Figure 20: Connexion - Cas de la failure**

# Sprint 2 : Gestion des paramètres

Ce deuxième sprint consiste à paramétrer les deux cartes embarquées par le biais de la gestion de leurs différents paramètres.

## Back log

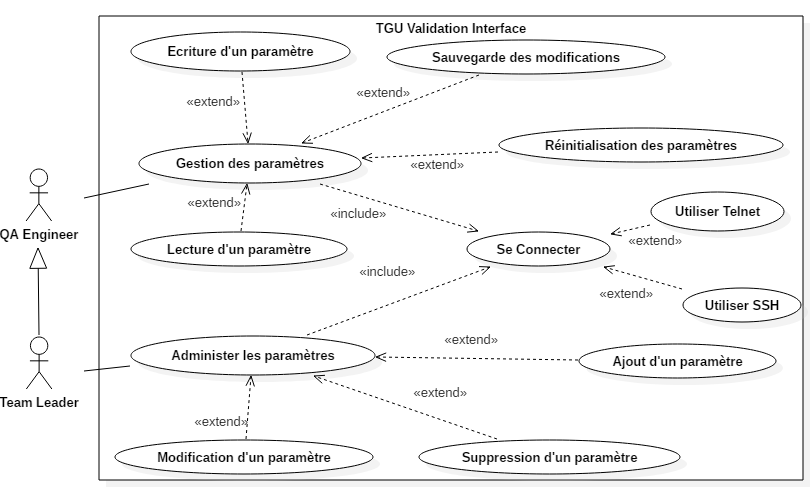
La planification de ce deuxième sprint est présentée par le tableau suivant :

|  |  |
| --- | --- |
| **User Story** | **Tâches** |
| Paramétrer | Lire la valeur d’un paramètre |
| Modifier la valeur d’un paramètre |
| Ajouter un paramètre |
| Modifier un paramètre |
| Supprimer un paramètre |
| Sauvegarder les modifications |
| Réinitialiser les paramètres |

**Tableau 4: Back log du sprint 2**

## Analyse

Le diagramme de cas d’utilisation suivant explique mieux le paramétrage des cartes.



**Figure 21: Gestion des paramètres**

Vers la fin de ce sprint, le QA Engineer et le Team Leader doivent avoir la possibilité d’apporter des modifications sur la valeur d’un paramètre ou sur le paramètre lui-même.

Le tableau suivant représente une description textuelle du scénario « Lire la valeur d’un paramètre » :

|  |  |
| --- | --- |
| Acteur | Utilisateur (QA Engineer ou Team Leader) |
| Résumé | Si l’utilisateur veut consulter la valeur d’un paramètre, il peut le faire à travers cette fonctionnalité |
| Précondition | Connexion déjà établie |
| Post-condition | Contenu d’un paramètre affiché |
| Scénario nominal | 1. L’utilisateur clique sur le bouton « Load » pour afficher les paramètres du calculateur 2. Le système affiche les paramètres dans un tableau 3. L’utilisateur choisit le paramètre et clique sur le bouton « Edit parameter » 4. Le Système affiche le paramètre avec sa valeur actuelle |
| Exceptions | -- |

**Tableau 5: Description textuelle de "Lire la valeur d'un paramètre"**

Le tableau suivant représente une description textuelle du scénario « Ecrire une valeur dans un paramètre » :

|  |  |
| --- | --- |
| Acteur | Utilisateur (QA Engineer ou Team Leader) |
| Résumé | Si l’utilisateur veut modifier la valeur d’un paramètre, il peut le faire à travers cette fonctionnalité |
| Précondition | Connexion déjà établie |
| Post-condition | Contenu d’un paramètre modifié |
| Scénario nominal | 1. L’utilisateur clique sur le bouton « Load » pour afficher les paramètres du calculateur 2. Le système affiche les paramètres dans un tableau 3. L’utilisateur choisit le paramètre et clique sur le bouton « Edit parameter » 4. Le Système affiche le paramètre avec sa valeur actuelle 5. L’utilisateur écrit la nouvelle valeur 6. L’utilisateur clique sur le bouton « Set Value » 7. Le système affecte la nouvelle valeur au paramètre |
| Exceptions | Le champ est vide  La nouvelle valeur est hors plage |

**Tableau 6: Description textuelle de "Ecrire une valeur dans un paramètre"**

Le tableau suivant représente une description textuelle du scénario « Sauvegarder les modifications » :

|  |  |
| --- | --- |
| Acteur | Utilisateur (QA Engineer ou Team Leader) |
| Résumé | Si l’utilisateur veut que les modifications faites soient persistantes, il peut le faire à travers cette fonctionnalité |
| Précondition | Connexion déjà établie |
| Post-condition | Modification de la valeur d’un paramètre sauvegardée |
| Scénario nominal | 1. L’utilisateur clique sur le bouton « Load » pour afficher les paramètres du calculateur 2. L’utilisateur modifie la valeur d’un ou plusieurs paramètres 3. L’utilisateur clique sur le bouton « Make the modification persistent » 4. Le Système enregistre tous les paramètres avec leurs nouvelles valeurs d’une manière persistante |
| Exceptions | Le champ est vide  La nouvelle valeur est hors plage |

**Tableau 7: Description textuelle de "Sauvegarder les modifications"**

Le tableau suivant représente une description textuelle du scénario « Réinitialiser les paramètres » :

|  |  |
| --- | --- |
| Acteur | Utilisateur (QA Engineer ou Team Leader) |
| Résumé | Si l’utilisateur veut réinitialiser les paramètres, il peut le faire à travers cette fonctionnalité |
| Précondition | Connexion déjà établie |
| Post-condition | Paramètres réinitialisés à leurs valeurs par défaut |
| Scénario nominal | 1. L’utilisateur clique sur le bouton « Reset » 2. Le Système réinitialise tous les paramètres à leurs valeurs par défaut |
| Exceptions | -- |

**Tableau 8: Description textuelle de "Réinitialiser les paramètres"**

Le tableau suivant représente une description textuelle du scénario « Ajouter un paramètre » :

|  |  |
| --- | --- |
| Acteur | Utilisateur (Team Leader) |
| Résumé | Si l’utilisateur veut ajouter un paramètre, il peut le faire à travers cette fonctionnalité |
| Précondition | Connexion déjà établie |
| Post-condition | Nouveau paramètre ajouté à la liste déjà existante |
| Scénario nominal | 1. L’utilisateur clique sur le bouton « Add » 2. Le système affiche une interface avec un champ de saisi de mot de passe 3. L’utilisateur saisit le mot de passe et clique sur le bouton « OK » 4. Le système affiche une interface contenant un formulaire à remplir 5. L’utilisateur remplit les champs du formulaire avec les données du nouveau paramètre et clique sur le bouton « Valider » 6. Le Système ajoute le nouveau paramètre à la liste des paramètres existants |
| Exceptions | Le mot de passe est incorrect  Le champ est vide |

**Tableau 9: Description textuelle de "Ajouter un paramètre"**

Le tableau suivant représente une description textuelle du scénario « Modifier un paramètre » :

|  |  |
| --- | --- |
| Acteur | Utilisateur (Team Leader) |
| Résumé | Si l’utilisateur veut modifier un paramètre, il peut le faire à travers cette fonctionnalité |
| Précondition | Connexion déjà établie |
| Post-condition | Paramètre modifié |
| Scénario nominal | 1. L’utilisateur clique sur le bouton « Load » pour afficher les paramètres du calculateur 2. Le système affiche la liste des paramètres existants 3. L’utilisateur choisit le paramètre à modifier 4. L’utilisateur clique sur le bouton « Modify » 5. Le système affiche une interface avec un champ de saisi de mot de passe 6. L’utilisateur saisit le mot de passe et clique sur le bouton « OK » 7. Le système affiche les informations relatives au paramètre sélectionné 8. L’utilisateur modifie les champs souhaités 9. L’utilisateur clique sur le bouton « Valider » 10. Le Système met à jour le paramètre |
| Exceptions | Le mot de passe est incorrect  Le champ est vide |

**Tableau 10: Description textuelle de "Modifier un paramètre"**

Le tableau suivant représente une description textuelle du scénario « Suprimer un paramètre » :

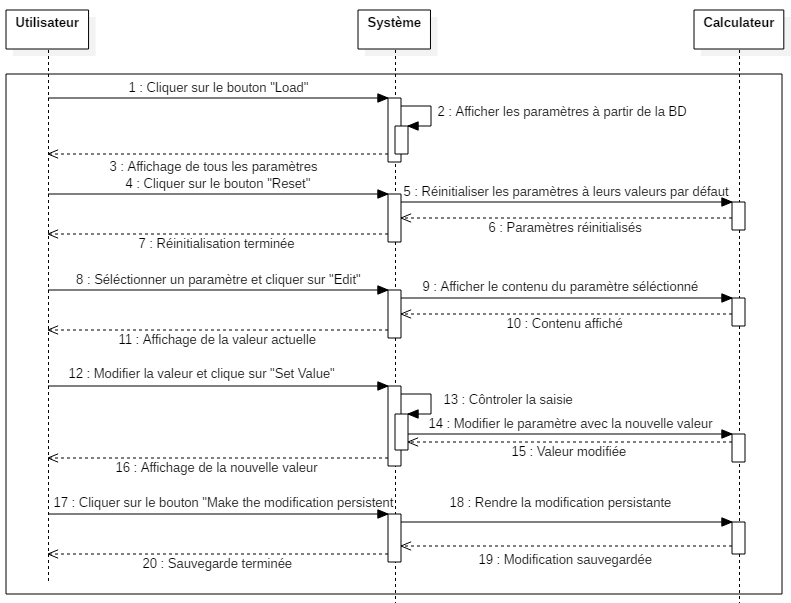
|  |  |
| --- | --- |
| Acteur | Utilisateur (Team Leader) |
| Résumé | Si l’utilisateur veut supprimer un paramètre, il peut le faire à travers cette fonctionnalité |
| Précondition | Connexion déjà établie |
| Post-condition | Paramètre supprimé |
| Scénario nominal | 1. L’utilisateur clique sur le bouton « Load » pour afficher les paramètres du calculateur 2. Le système affiche la liste des paramètres existants 3. L’utilisateur sélectionne le paramètre qu’il veut supprimer 4. L’utilisateur clique sur le bouton « Delete » 5. Le système affiche une interface avec un champ de saisi de mot de passe 6. L’utilisateur saisit le mot de passe et clique sur le bouton « OK » 7. Le système affiche un message de confirmation demandant si l’utilisateur veut réellement supprimer le paramètre 8. L’utilisateur confirme la suppression du paramètre 9. Le Système supprime le paramètre |
| Exceptions | Le mot de passe est incorrect  Le champ est vide |

**Tableau 11: Description textuelle de "Supprimer un paramètre"**

## Conception

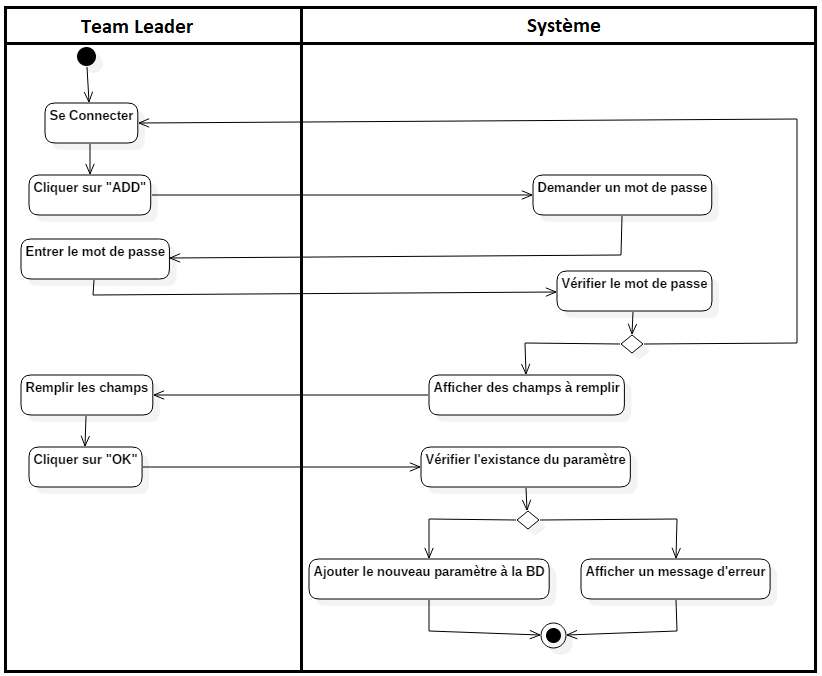
Pour ne pas surcharger le rapport, nous allons combiner quelques cas d’utilisation dans un seul diagramme de séquence et nous présentons les autres dans l’annexe.

La figure suivante représente le déroulement des cas d’utilisation « Réinitialiser, lire, écriture et sauvegarder la modification d’un paramètre »



**Figure 22: Diagramme de séquence de « Réinitialiser, lire, écriture et sauvegarder la modification d’un paramètre"**

La figure suivante représente le déroulement du scénario « Ajouter un paramètre »

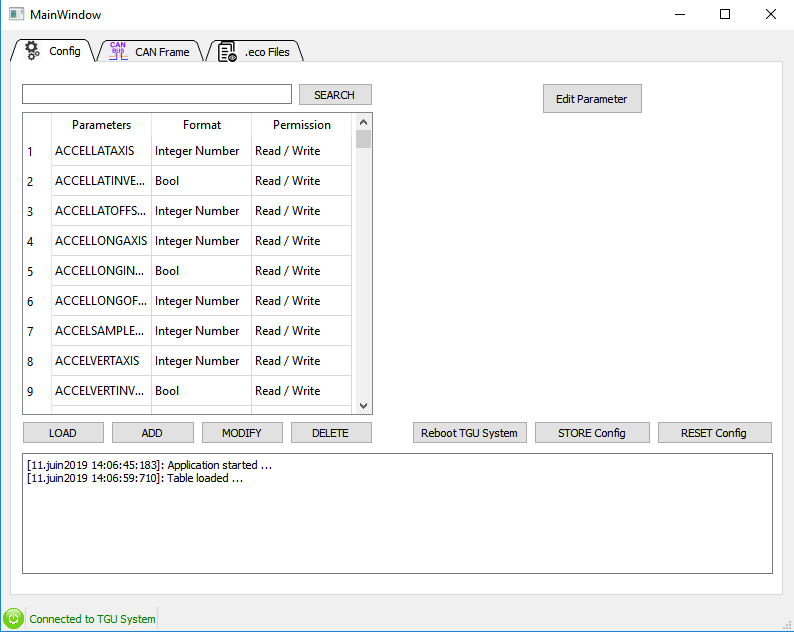


**Figure 23: Diagramme d'activité de "Ajouter un paramètre"**

## Réalisation

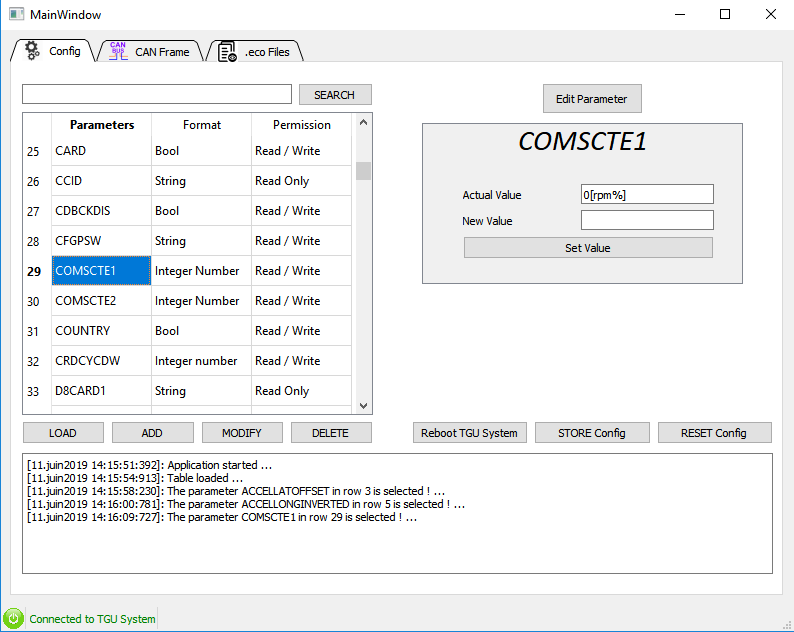
A la fin de ce sprint, nous avons pu livrer à notre Product Owner une application qui permet de paramétrer nos calculateurs à travers la gestion des paramètres. Les figures suivantes représentent le travail réalisé à ce propos.

Pour afficher tous les paramètres, il faut cliquer sur le bouton « Load »

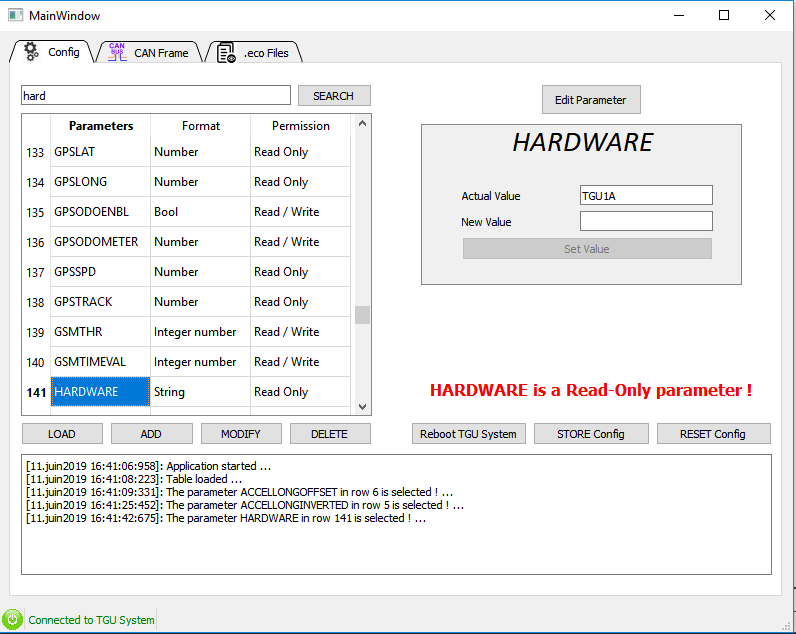


**Figure 24: Affichage de tous les paramètres**

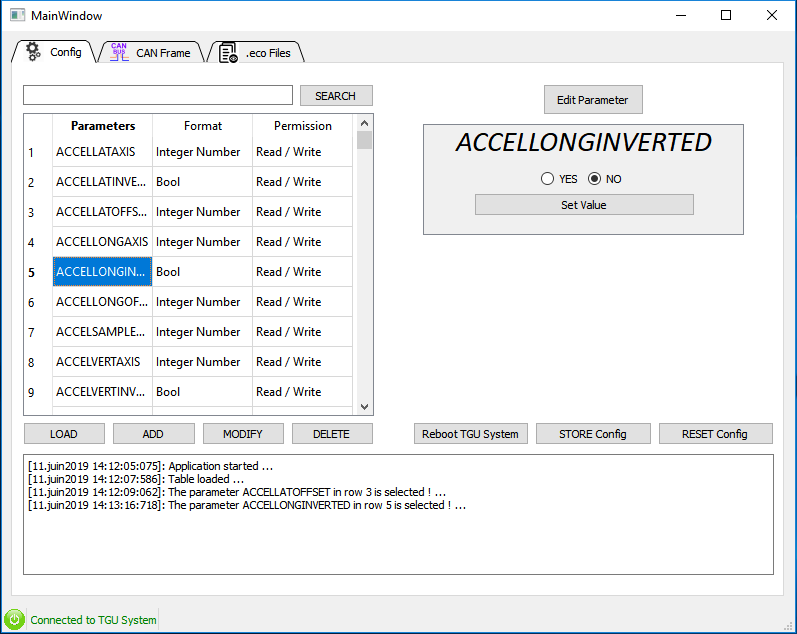
Un double clic sur le nom du paramètre ou un seul clic sur le bouton « Edit Parameter » suffit pour afficher son contenu.



**Figure 25: Affichage du contenu d'un paramètre (En mode lecture et écriture)**

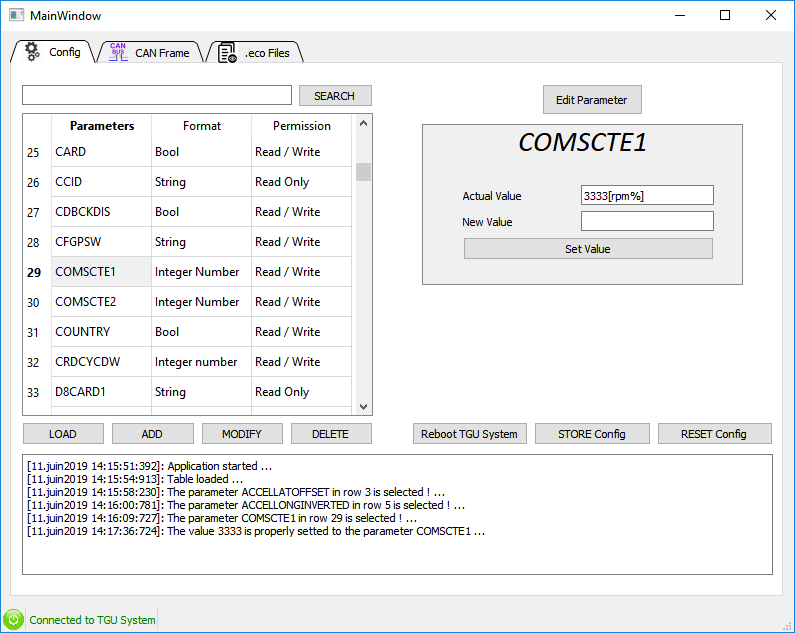


**Figure 26: Affichage du contenu d'un paramètre (En mode lecture seule)**



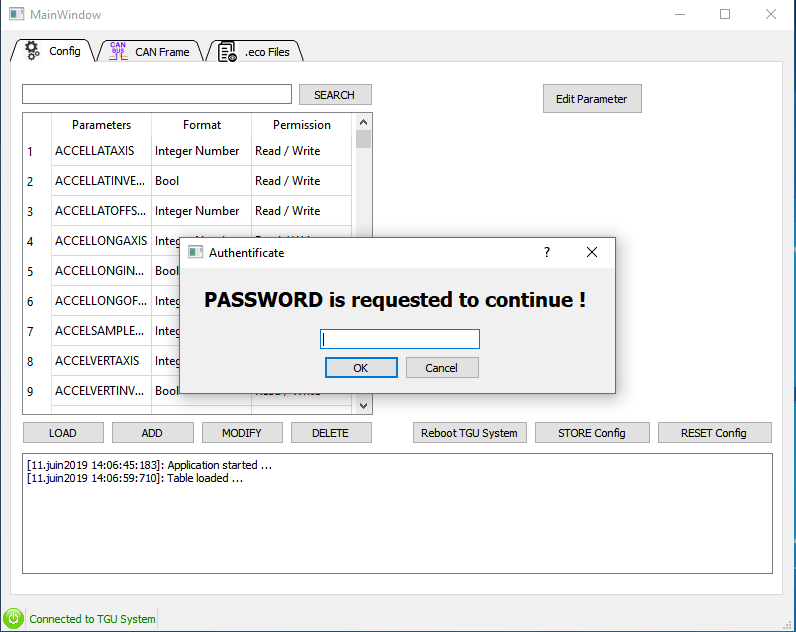
**Figure 27: Affichage du contenu d'un paramètre booléen**

Pour la modification de la valeur des paramètres, l’utilisateur doit introduire la nouvelle valeur dans le champ « New Value » et clique par la suite sur le bouton « Set Value ».



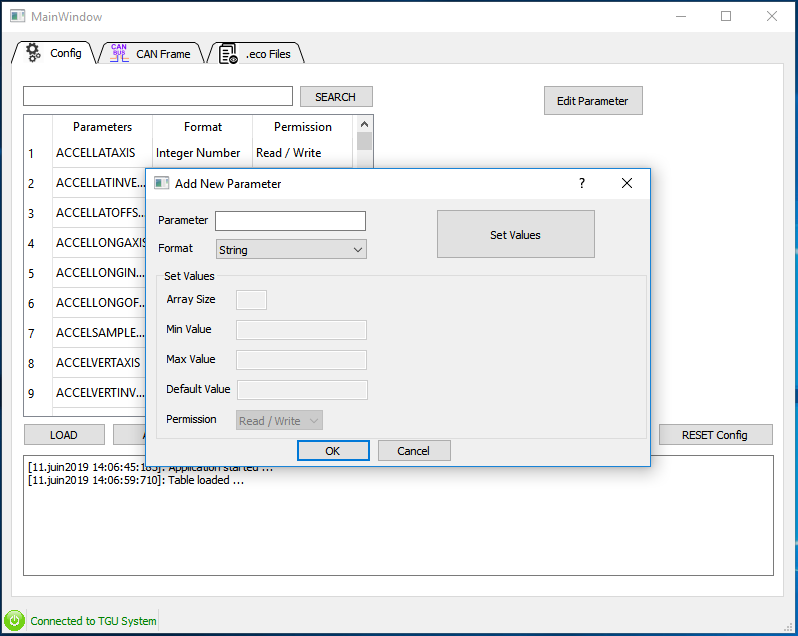
**Figure 28: Modification de la valeur actuelle d'un paramètre**

Pour ajouter, modifier ou supprimer un paramètre, il faut passer par une autre interface qui contrôle l’accès par le biais d’un mot de passe valide.



**Figure 29: Interface pour contrôler l'accès**

Après avoir entré le bon mot de passe, les actions d’ajout et de modification d’un paramètre sont faites à travers l’interface suivante.



**Figure 30: Interface pour ajouter ou modifier un paramètre**

# Sprint 3 : Gestion des trames

Ce troisième sprint consiste à simuler l’envoi des trames CAN afin de voir le comportement du calculateur TGU.

## Back log

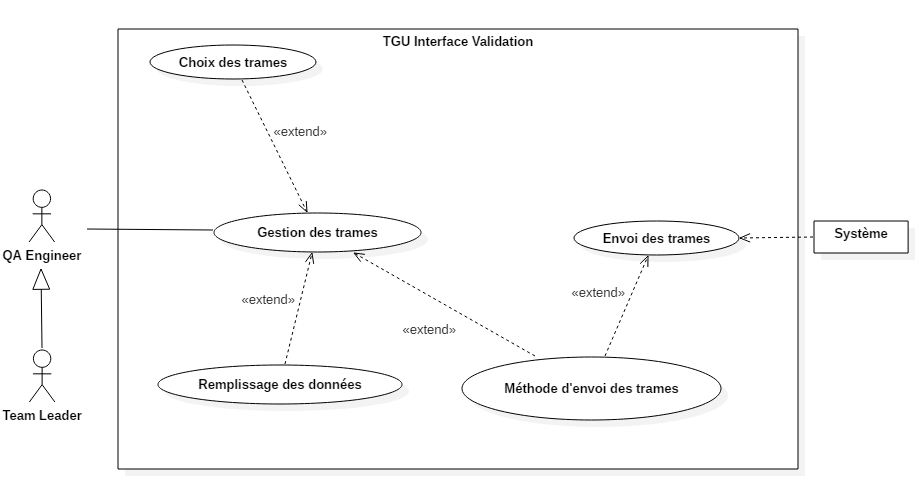
La planification de ce troisième sprint est présentée par le tableau suivant :

|  |  |
| --- | --- |
| **User Story** | **Tâches** |
| Gérer les trames | Choisir la trame |
| Remplir les données de la trame |
| Envoyer la trame |

**Tableau 12: Back log du sprint 3**

## Analyse

Le diagramme de cas d’utilisation suivant explique mieux la gestion des trames CAN.

**Figure 31: Gestion des trames**

Vers la fin de ce sprint, le QA Engineer et le Team Leader doivent avoir la possibilité de choisir la trame à envoyer et sa méthode d’envoi. Il existe plusieurs combinaisons pour décrire ce cas d’utilisation et pour ne pas surcharger le rapport, on va citer les 4 scénarios suivants :

* Envoyer une trame une seule fois
* Envoyer une trame plusieurs fois
* Envoyer un ensemble de trames
* Envoyer une trame avec le protocole BAM

Le tableau suivant représente une description textuelle du scénario « Envoyer une trame une seule fois » :

|  |  |
| --- | --- |
| Acteur | Utilisateur (QA Engineer ou Team Leader) |
| Résumé | Si l’utilisateur veut envoyer une trame une seule fois, il peut le faire à travers cette fonctionnalité |
| Précondition | Connexion déjà établie |
| Post-condition | Trame envoyée une seule fois |
| Scénario nominal | 1. L’utilisateur choisit l’onglet « CAN frame » 2. Le système affiche une interface permettant de choisir les trames et les envoyer 3. L’utilisateur choisit la trame depuis la liste déroulante et saisit sa priorité ainsi que sa source adresse 4. L’utilisateur remplit les données de la trame dans les champs correspondants 5. L’utilisateur choisit l’option « Send once » qui permet d’envoyer la trame une seule fois 6. L’utilisateur clique sur le bouton « Generate » 7. Le système envoie la trame choisie avec les mêmes données pré remplies |
| Exceptions | -- |

**Tableau 13: Description textuelle de "Envoyer une trame une seule fois"**

Le tableau suivant représente une description textuelle du scénario « Envoyer une trame plusieurs fois » :

|  |  |
| --- | --- |
| Acteur | Utilisateur (QA Engineer ou Team Leader) |
| Résumé | Si l’utilisateur veut envoyer une trame plusieurs fois, il peut le faire à travers cette fonctionnalité |
| Précondition | Connexion déjà établie |
| Post-condition | Trame envoyée N fois avec X cycle |
| Scénario nominal | 1. L’utilisateur choisit l’onglet « CAN frame » 2. Le système affiche une interface permettant de choisir les trames et les envoyer 3. L’utilisateur choisit la trame depuis la liste déroulante et saisit sa priorité ainsi que sa source adresse 4. Le système affiche les signaux qui correspondent à la trame choisie dans le champ « Choose SPN » 5. L’utilisateur Choisi un SPN 6. L’utilisateur remplit le champ « Set Value » par la valeur souhaitée 7. L’utilisateur clique sur le bouton « Set SPN Value » 8. Le système remplit automatiquement les données de la trame avec la valeur saisie 9. L’utilisateur choisit l’option « Send » en remplissant le champ « Time(s) » avec le nombre de fois qu’on souhaite envoyer la trame et le champ « Cycle (ms) » avec la périodicité d’envoi 10. L’utilisateur clique sur le bouton « Generate » 11. Le système envoie la trame choisie avec les mêmes données pré remplies |
| Exceptions | -- |

**Tableau 14: Description textuelle de "Envoyer une trame plusieurs fois"**

Le tableau suivant représente une description textuelle du scénario « Envoyer un ensemble de trames » :

|  |  |
| --- | --- |
| Acteur | Utilisateur (QA Engineer ou Team Leader) |
| Résumé | Si l’utilisateur veut envoyer un ensemble de trames, il peut le faire à travers cette fonctionnalité |
| Précondition | Connexion déjà établie |
| Post-condition | Trames envoyées |
| Scénario nominal | 1. L’utilisateur choisit l’onglet « CAN frame » 2. Le système affiche une interface permettant de choisir les trames et les envoyer 3. L’utilisateur choisit la trame depuis la liste déroulante et saisit sa priorité ainsi que sa source adresse 4. Le système affiche les signaux qui correspondent à la trame choisie dans le champ « Choose SPN » 5. L’utilisateur Choisi un SPN 6. L’utilisateur remplit le champ « Set Value » par la valeur souhaitée 7. Le système remplit automatiquement les données de la trame avec la valeur saisie 8. L’utilisateur clique sur le bouton « Add » 9. Le système ajoute la trame dans une liste à envoyer 10. L’utilisateur choisit la trame depuis la liste déroulante et saisit sa priorité ainsi que sa source adresse et refait les actions précédentes 11. L’utilisateur clique sur le bouton « Generate » 12. Le système envoie les deux trames choisies avec les mêmes données pré remplies |
| Exceptions | -- |

**Tableau 15: Description textuelle de "Envoyer un ensemble de trame"**

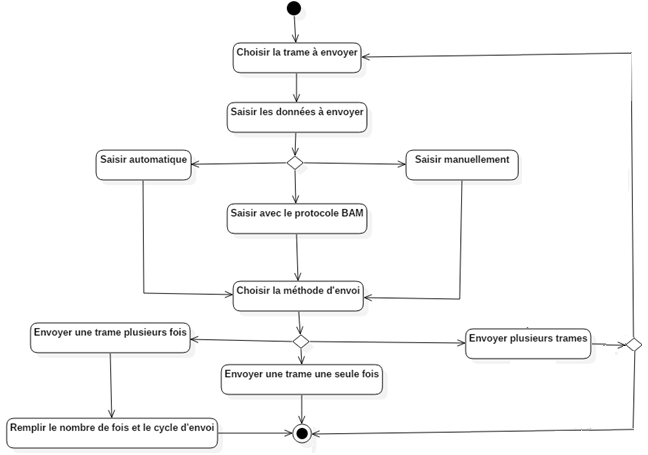
Le tableau suivant représente une description textuelle du scénario « Envoyer une trame avec le protocole BAM » :

|  |  |
| --- | --- |
| Acteur | Utilisateur (QA Engineer ou Team Leader) |
| Résumé | Si l’utilisateur veut envoyer une trame avec le protocole BAM, il peut le faire à travers cette fonctionnalité |
| Précondition | Connexion déjà établie |
| Post-condition | Trame envoyée avec protocole BAM |
| Scénario nominal | 1. L’utilisateur choisit l’onglet « CAN frame » 2. Le système affiche une interface permettant de choisir les trames et les envoyer 3. L’utilisateur choisit la trame depuis la liste déroulante et saisit sa priorité ainsi que sa source adresse 4. L’utilisateur choisit l’option d’envoi avec le protocole BAM 5. L’utilisateur saisit la longueur de données à envoyer 6. L’utilisateur saisit les données de la trame à envoyer 7. L’utilisateur clique sur le bouton « Generate » 8. Le système envoie la trame choisie avec le protocole BAM avec les mêmes données pré remplies |

**Tableau 16: Description textuelle de "Envoyer une trame avec le protocole BAM"**

## Conception

La figure suivante représente le diagramme d’activité pour ce sprint. Elle détaille ses fonctionnalités et leur interconnexion.

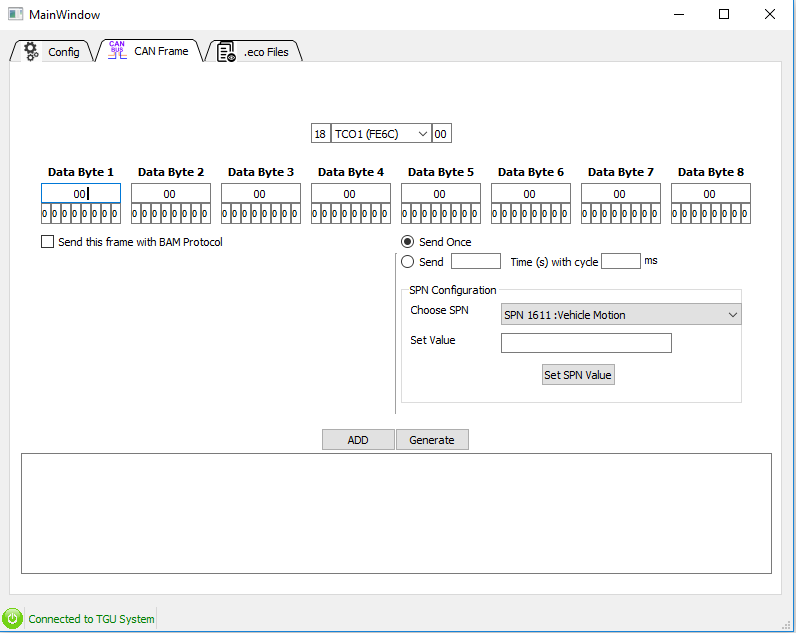


**Figure 32:Diagramme d'activité du sprint "Gestion des trames**"

## Réalisation

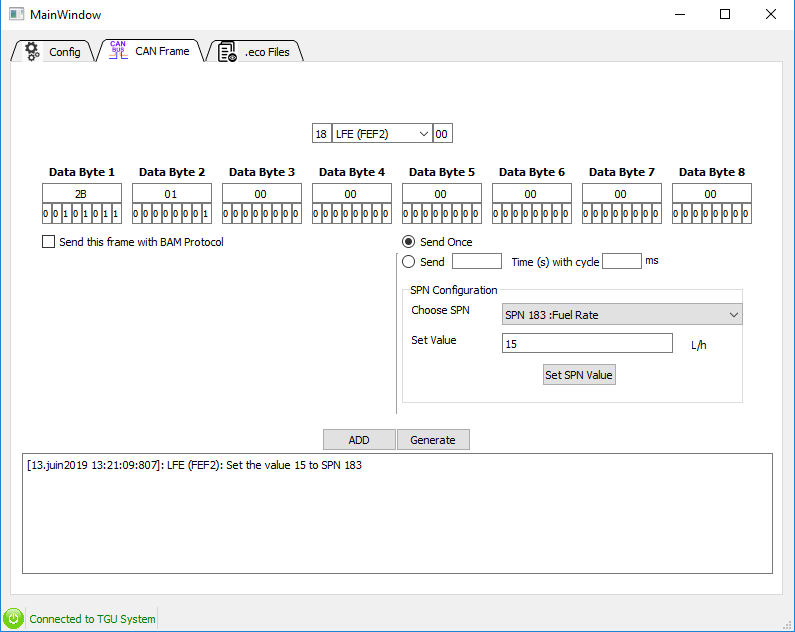
A la fin de ce sprint, nous avons pu livrer à notre Product Owner une application qui permet de simuler l’envoi des trames CAN à travers la gestion des trames. Les figures suivantes représentent le travail réalisé à ce propos.

Pour remplir manuellement les données de la trame, l’utilisateur choisit l’onglet « CAN frame », sélectionne la trame depuis la liste déroulante, puis remplit manuellement les données dans les champs correspondants.



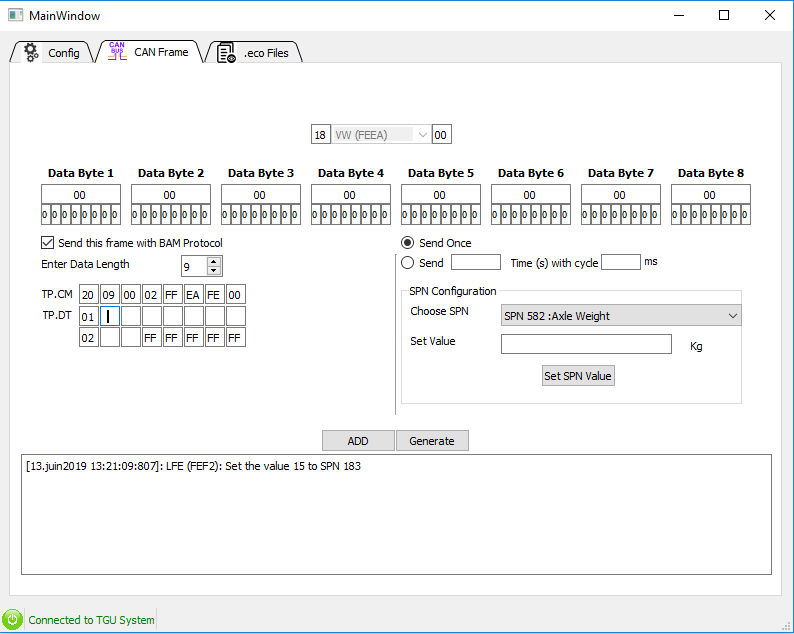
**Figure 33: Remplissage manuel des données**

Pour que les données soient saisies automatiquement, il faut sélectionner la trame depuis la liste déroulante, choisir le SPN correspondant à partir de la liste déroulante du champ « Choose SPN », remplir le champ « Set value » par la valeur souhaitée en décimal, puis cliquer sur le bouton « Set SPN Value ».



**Figure 34: Remplissage automatique des données**

Si l’utilisateur veut envoyer une trame avec une longueur qui dépasse 8 octets, il utilise le protocole BAM comme l’indique la figure suivante.

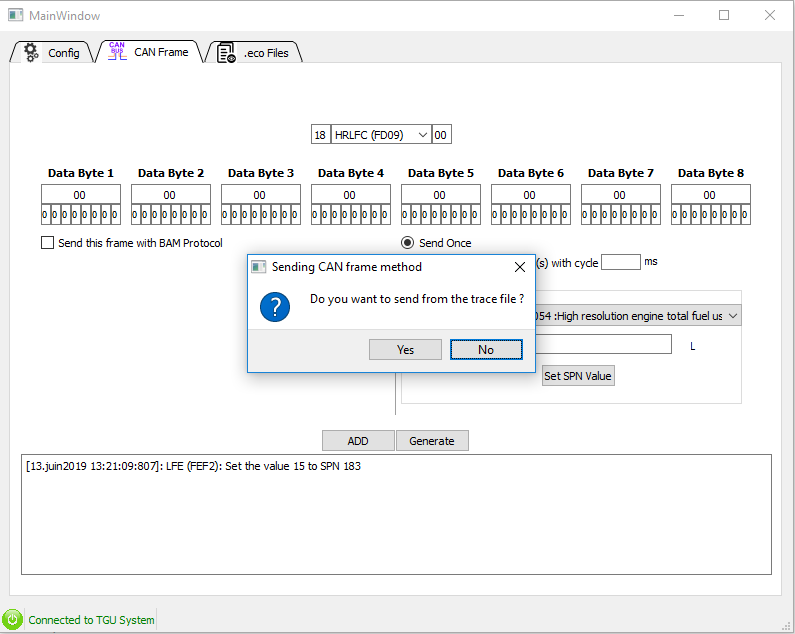
****

**Figure 35: Remplissage des données avec BAM**

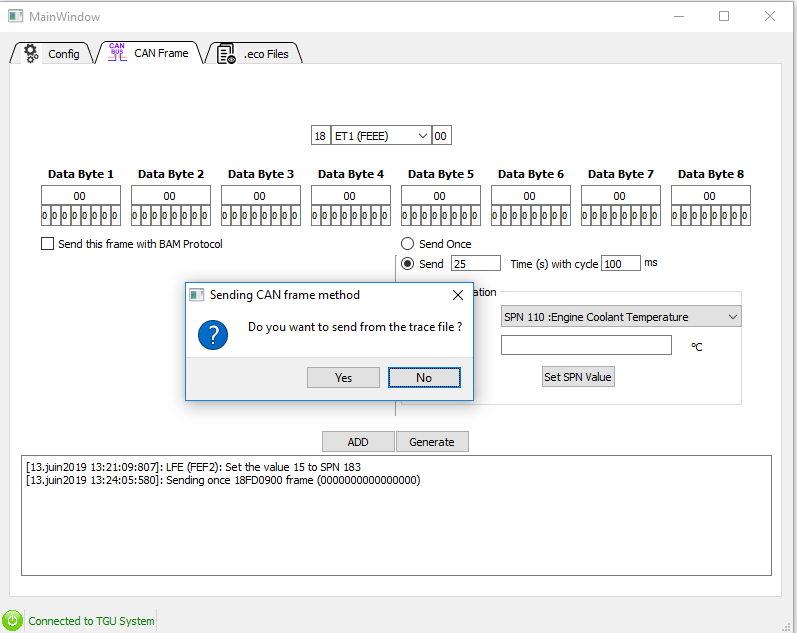
Après avoir terminé le remplissage des données par l’une des méthodes mentionnées ci-dessus, l’utilisateur peut envoyer les trames CAN par deux façons :

1. Envoyer une seule trame

L’utilisateur peut envoyer la trame une seule fois en choisissant l’option « Send Once » ou plusieurs fois en choisissant l’option « Send » avec le nombre et la fréquence d’envoi.



**Figure 36: Envoyer une seule trame une seule fois**



**Figure 37: Envoyer une seule trame plusieurs fois**

1. Envoyer plusieurs trames

Pour envoyer plusieurs trames, il faut cliquer sur le bouton « ADD », ajouter le nombre de trames souhaité, cliquer sur le bouton « Generate » puis cliquer sur le bouton « Yes » du popup qui s’affiche.

**Figure 38: Envoyer plusieurs trames**

# Conclusion

Tout au long de ce chapitre, nous avons présenté les trois premiers sprints, leurs back logs ainsi qu’une explication de toutes les phases d’analyse, de conception et de réalisation. Nous avons entamé les trois derniers sprints.

# Chapitre 5 : Conception détaillée du projet – Release 2

*« Chaque bonne réalisation, grande ou petite,*

*connait ses périodes de corvées et de triomphes; un début,*

*un combat et une victoire. »*

# Introduction

Dans ce chapitre les trois derniers sprints vont être décrits par leurs back logs et expliqués durant toutes les phases d’analyse, de conception et de réalisation. A la fin de chaque sprint, nous aurons un produit stable représenté par des captures d’écran.

# Sprint 4 : Gestion des fichiers

Ce quatrième sprint consiste à traduire le contenu des fichiers .eco générés afin de les rendre lisibles et compréhensibles.

## Back log

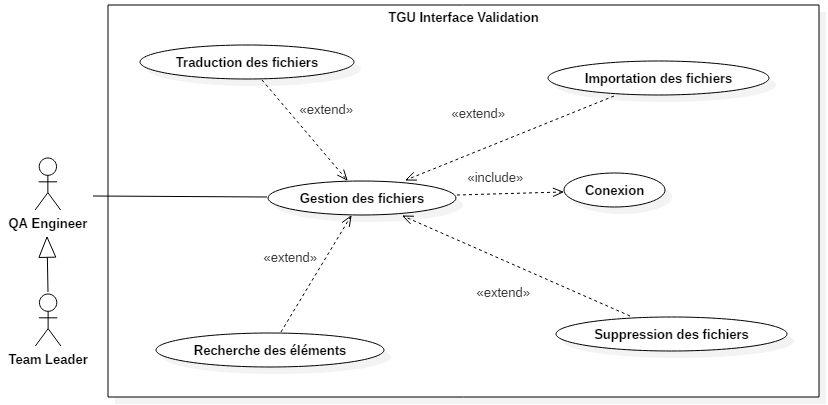
La planification de ce quatrième sprint est présentée par le tableau suivant :

|  |  |
| --- | --- |
| **User Story** | **Tâches** |
| Gérer les fichiers | Traduction du contenu de fichier |
| Recherche d’élément dans le fichier |
| Importation des fichiers |
| Suppression des fichiers |

**Tableau 17: Back log du sprint 4**

## Analyse

Le diagramme de cas d’utilisation suivant explique mieux la gestion des fichiers.



**Figure 39: Gestion des fichiers**

Vers la fin de ce sprint, le QA Engineer et le Team Leader doivent avoir la possibilité d’importer les fichiers depuis la carte embarquée, de les supprimer, de les traduire pour faciliter leurs lectures comme ils peuvent effectuer une recherche des éléments dans le fichier.

Le tableau suivant représente une description textuelle du scénario « Importer les fichiers » :

|  |  |
| --- | --- |
| Acteur | Utilisateur (QA Engineer ou Team Leader) |
| Résumé | Si l’utilisateur veut importer les fichiers existants dans la carte embarquée, il peut le faire à travers cette fonctionnalité |
| Précondition | Fichier .eco généré dans la carte |
| Post-condition | Fichier importé |
| Scénario nominal | 1. L’utilisateur choisit l’onglet « .eco files » 2. Le système affiche une interface permettant d’attacher le fichier .eco 3. L’utilisateur clique sur le bouton « Import Files » 4. Le système récupère les fichiers depuis la carte et demande à l’utilisateur de choisir l’emplacement souhaité pour les enregistrer. 5. L’utilisateur choisit l’emplacement. 6. Le système enregistre les fichiers dans l’emplacement choisi |
| Exceptions | -- |

**Tableau 18: Description textuelle du "Imprter les fichiers"**

Le tableau suivant représente une description textuelle du scénario « Supprimer les fichiers » :

|  |  |
| --- | --- |
| Acteur | Utilisateur (QA Engineer ou Team Leader) |
| Résumé | Si l’utilisateur veut supprimer les fichiers existants dans la carte embarquée, il peut le faire à travers cette fonctionnalité |
| Précondition | Fichier .eco généré dans la carte |
| Post-condition | Fichier supprimé |
| Scénario nominal | 1. L’utilisateur choisit l’onglet « .eco files » 2. Le système affiche une interface permettant d’attacher le fichier .eco 3. L’utilisateur clique sur le bouton « Delete Files » 4. Le système supprime les fichiers existants dans la carte. |
| Exceptions | -- |

**Tableau 19: Description textuelle du "Supprimer les fichiers"**

Le tableau suivant représente une description textuelle du scénario « Traduire le contenu du fichier » :

|  |  |
| --- | --- |
| Acteur | Utilisateur (QA Engineer ou Team Leader) |
| Résumé | Si l’utilisateur veut traduire le contenu du fichier .eco, il peut le faire à travers cette fonctionnalité |
| Précondition | Fichier .eco généré |
| Post-condition | Fichier traduit |
| Scénario nominal | 1. L’utilisateur choisit l’onglet « .eco files » 2. Le système affiche une interface permettant d’attacher le fichier .eco 3. L’utilisateur clique sur le bouton « Open file » 4. Le système affiche un explorateur permettant de sélectionner le fichier à traduire 5. L’utilisateur choisit le fichier 6. Le système affiche le contenu traduit du fichier dans l’interface |
| Exceptions | -- |

**Tableau 20: Description textuelle du "Traduire le contenu du fichier"**

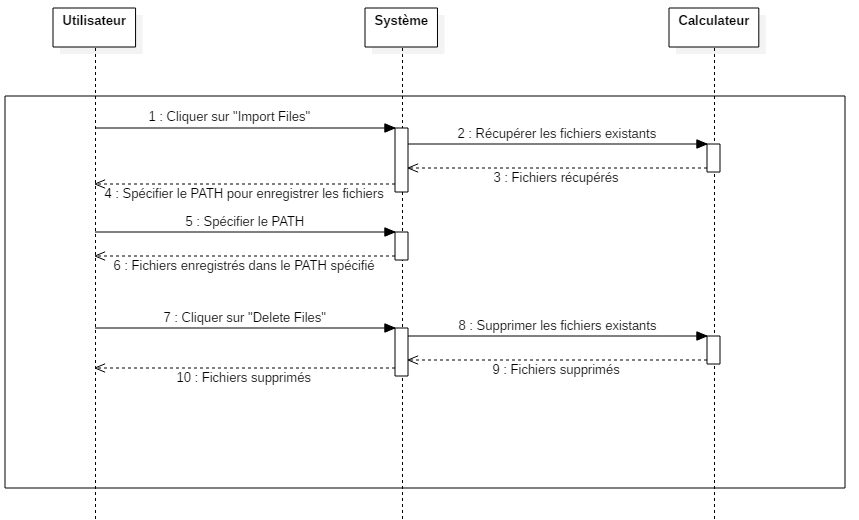
Le tableau suivant représente une description textuelle du scénario « Rechercher un élément dans le fichier » :

|  |  |
| --- | --- |
| Acteur | Utilisateur (QA Engineer ou Team Leader) |
| Résumé | Si l’utilisateur veut rechercher un élément dans le fichier .eco, il peut le faire à travers cette fonctionnalité |
| Précondition | Fichier .eco traduit |
| Post-condition | Recherche effectuée |
| Scénario nominal | 1. L’utilisateur saisit l’élément à rechercher dans le champ de recherche 2. L’utilisateur clique sur le bouton « > » 3. Le système effectue la recherche. Si l’élément est trouvé, il le sélectionne. Sinon, il affiche un message indiquant l’inexistence de l’élément. |
| Exceptions | Champ vide |

**Tableau 21: Description textuelle du "Rechercher un élément dans le fichier "**

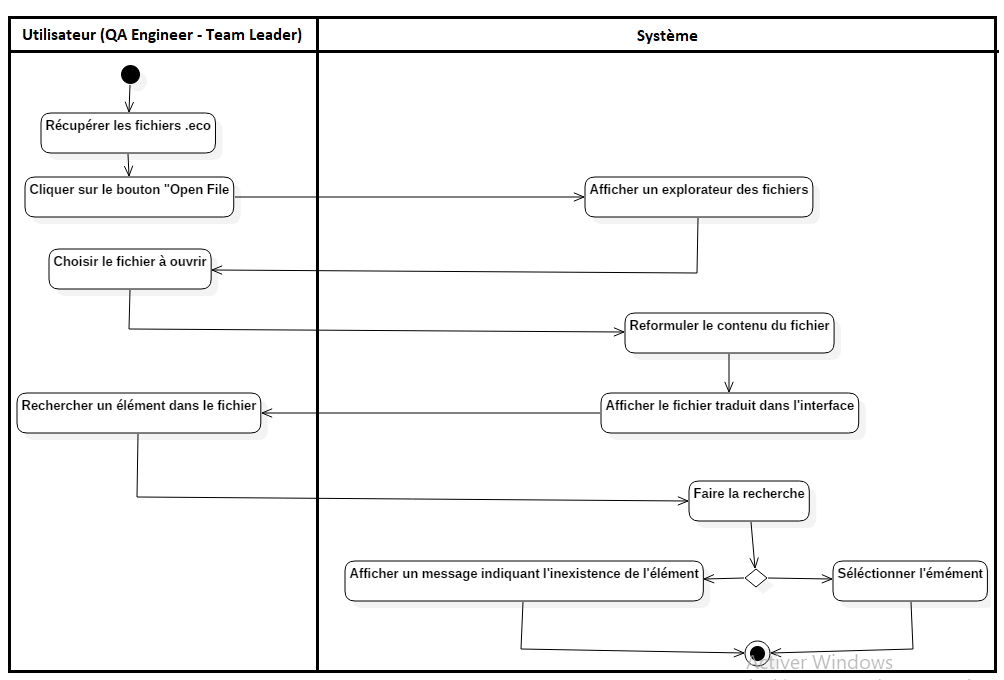
## Conception

Le diagramme de séquence suivant représente les parties d’importation et de suppression des fichiers de ce quatrième sprint.



**Figure 40: Diagramme de séquence du "Importer et Supprimer les fichiers"**

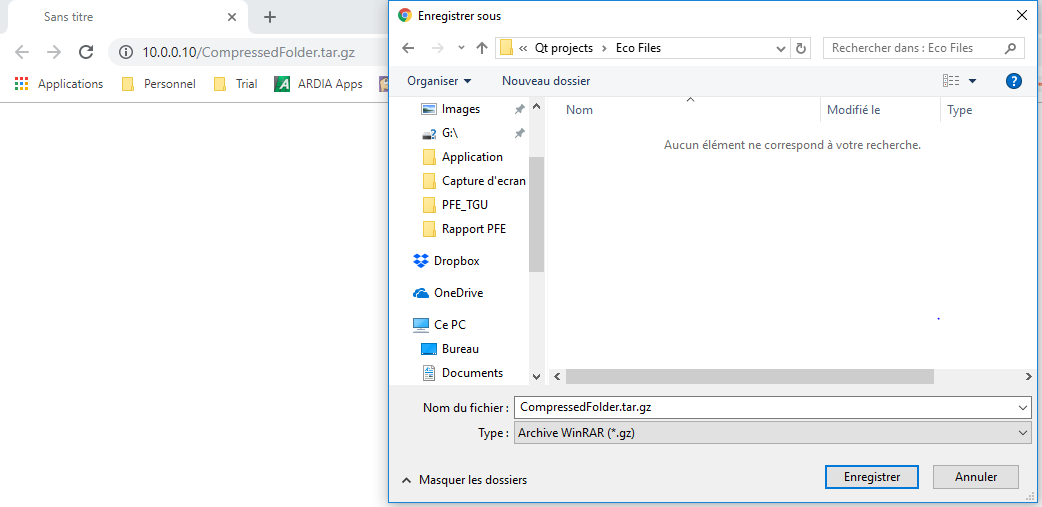
Le diagramme d’activité représenté par la figure suivante détaille les actions de traduction du fichier et de la recherche des éléments.



**Figure 41: Diagramme d'activité du sprint "Gestion des fichiers"**

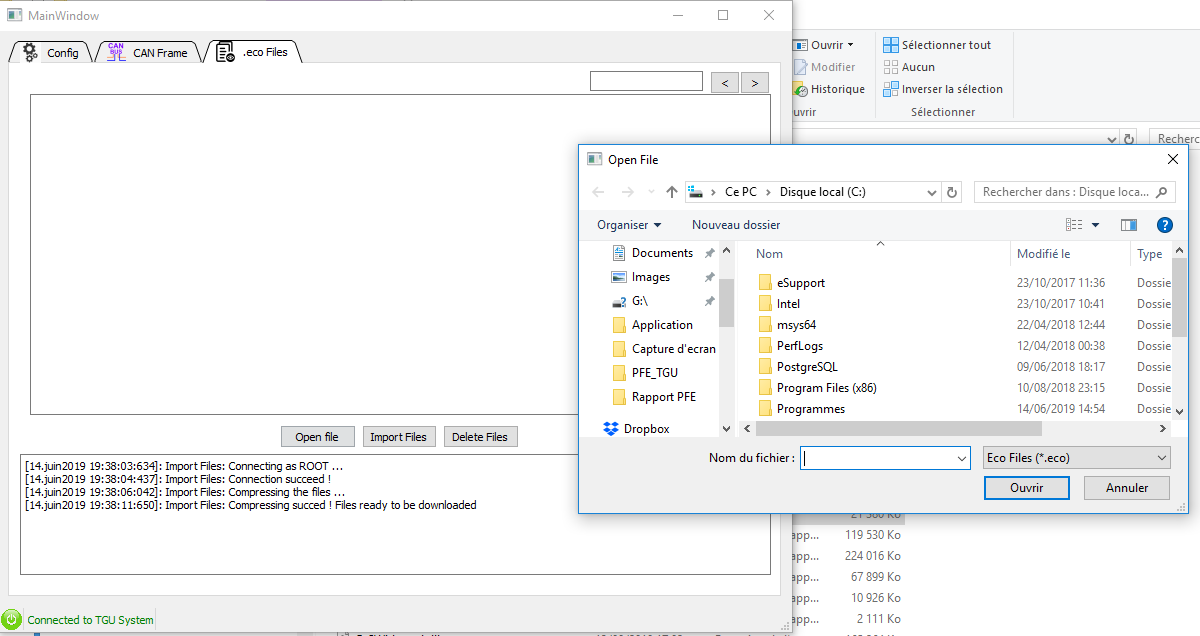
## Réalisation

Lorsque l’utilisateur clique sur le bouton « Import Files », l’interface suivante s’affiche pour lui demander de spécifier l’endroit où il souhaite enregistrer les fichiers.



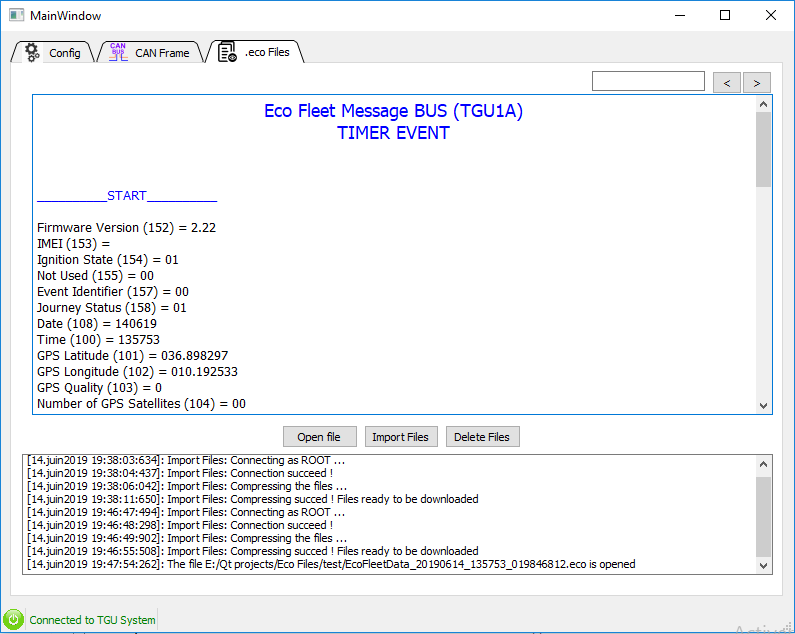
**Figure 42: Enregistrer les fichiers**

La figure suivante représente l’interface affichée après avoir cliqué sur le bouton « Open file » de l’onglet « .eco files ».



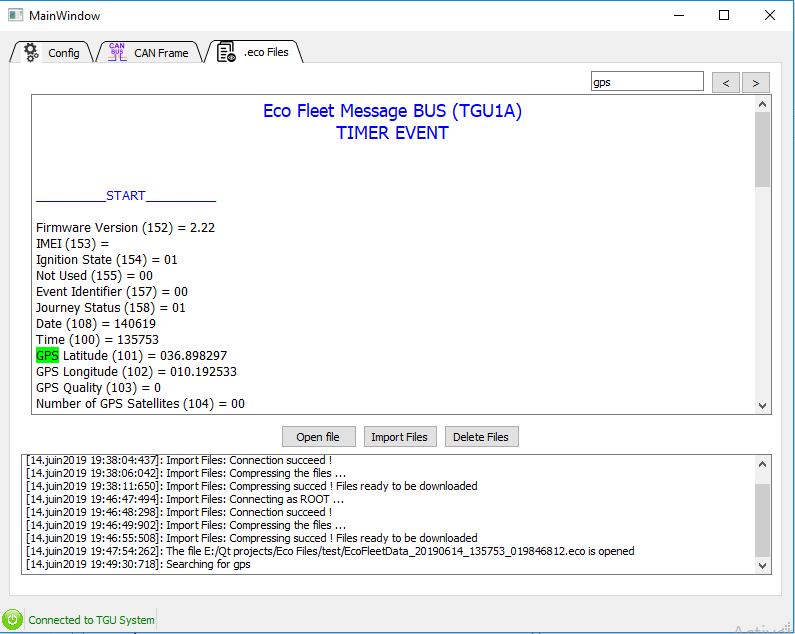
**Figure 43: Attacher un fichier**

Le fichier traduit sera affiché dans l’interface comme indique la figure ci-dessous.



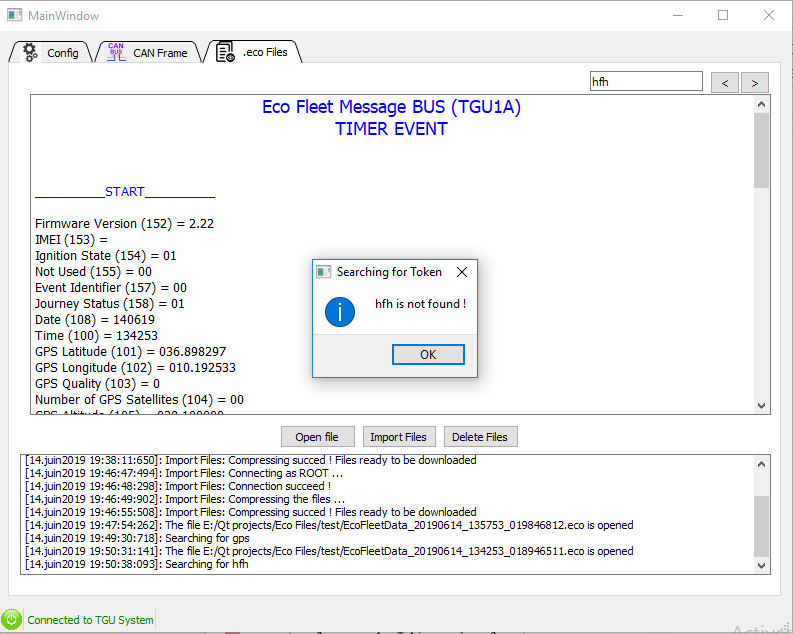
**Figure 44: Fichier traduit**

Lors de la recherche d’un élément, il sera sélectionné s’il est présent dans le fichier comme indiqué dans la figure suivante.



**Figure 45: Elément trouvé**

Sinon, le message suivant est affiché.



**Figure 46: Elément non trouvé**

# Sprint 5 : Génération des logs

Ce cinquième sprint consiste à générer des logs durant l’utilisation de l’application et les transformer en des fichiers après sa fermeture.

## Back log

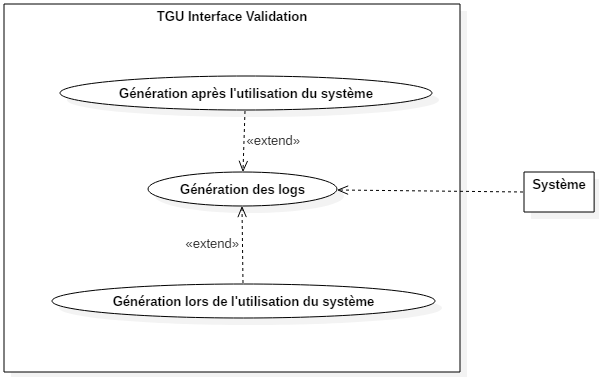
La planification de ce cinquième sprint est présentée par le tableau suivant :

|  |  |
| --- | --- |
| **User Story** | **Tâches** |
| Générer les logs | Générer les logs lors de l’utilisation de l’application |
| Générer les logs après la fermeture de l’application |

**Tableau 22: Back log du sprint 5**

## Analyse

Le diagramme de cas d’utilisation suivant explique mieux la gestion des logs



**Figure 47: Génération des logs**

Vers la fin de ce sprint, le système doit générer des logs lors et après l’utilisation de l’application.

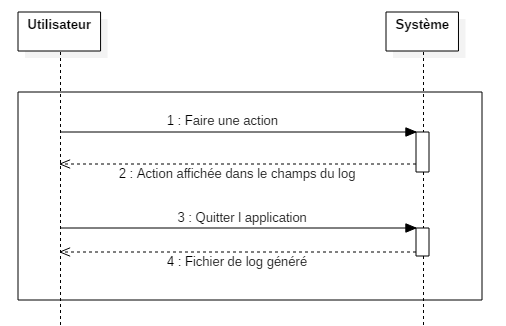
Le tableau suivant représente une description textuelle du scénario « Générer les logs » :

|  |  |
| --- | --- |
| Acteur | Système |
| Résumé | Le système génère les logs à travers cette fonctionnalité |
| Précondition | Application fermée |
| Post-condition | Logs générés |
| Scénario nominal | 1. L’utilisateur lance l’application 2. L’utilisateur fait des actions 3. Les actions faites s’affichent en temps réel dans le champ des logs 4. L’utilisateur ferme l’application 5. Le système génère un fichier texte contenant l’historique de toutes les actions effectuées par l’utilisateur |
| Exceptions | -- |

**Tableau 23: Description textuelle du "Générer les logs"**

## Conception

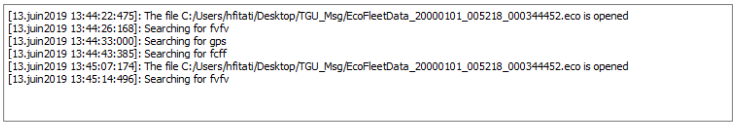
Le diagramme de séquence, ci-dessous, détaille la génération des logs par notre système.



**Figure 48: Diagramme de séquence du "Génération des logs"**

## Réalisation

La figure suivante présente un exemple de log généré par le système lors de l’utilisation de l’application.

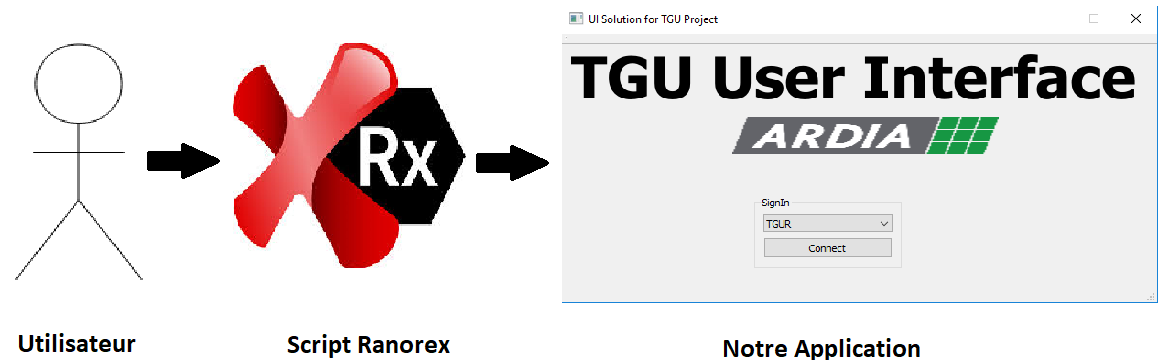


**Figure 49: Exemple de log généré**

# Sprint 6 : Automatisation d’un cas de test

L’objectif de ce sixième sprint est de prouver que notre application répond au besoin du projet, qui est l’automatisation des cas de tests des plans de validation.

L’utilisateur de notre application est appeler à lancer seulement les scripts Ranorex qui les développe et qui utilise notre application pour exécuter les tests de validation comme le montre la figure suivante.



**Figure 50: Automatisation des tests**

## Cas de test :

Une exigence dans la spécification fonctionnelle du projet TGU indique que si le conducteur dépasse une vitesse « x » pour une période « y », avec x et y sont des paramètres à configurer préalablement, le calculateur TGU doit identifier cet événement par la génération d’un fichier à ce propos lorsque la vitesse de la véhicule soit inférieur à « x ». Pour valider cette exigence, 3 cas de tests sont nécessaires :

|  |  |  |  |
| --- | --- | --- | --- |
| **Cas de test** | **Condition initiale** | **Procédure de passage** | **Résultat attendu** |
| 1er cas | X = 80 km/h ;  Y = 6 secondes | Vitesse > 80 ;  Période > 6 | Un événement est généré |
| 2ème cas | X = 80 km/h ;  Y = 6 secondes | Vitesse > 80 ;  Période < 6 | Pas d’événement généré |
| 3ème cas | X = 80 km/h ;  Y = 6 secondes | Vitesse < 80 ;  Période > 6 | Pas d’événement généré |

**Tableau 24: Cas des tests possibles pour l'exigence "Excès de vitesse"**

## Script Ranorex :

Pour le développement du script Ranorex, nous avons choisi le 1er cas de test vu que son exécution nécessite le passage de tous les sprints que nous avons réalisés.

Pour ce fait, voici les étapes que notre script les suits :

1. Lancer l’application TGU USER INTERFACE
2. Choisir le type de la carte à utiliser
3. Vérifier l’établissement de la connexion
4. Modifier les paramètres ECOSPEED et ECOSPDTHR respectivement par 80 et 6000.
5. Vérifier que la réussite de la modification
6. Choisir l’onglet CAN Frame
7. Choisir la trame CCVS depuis la liste déroulante des trames
8. Choisir le SPN 96 depuis la liste déroulante des SPN
9. Saisir la valeur 81 dans le champ SPN Value
10. Cliquer sur le bouton Set SPN Value
11. Vérifier la saisie automatique de la valeur entrée
12. Cocher l’option Send Many en introduisant la valeur 70 dans le champ correspond au nombre des trames et 100 dans le champ correspond au cycle
13. Cliquer sur le bouton ADD
14. Saisir la valeur 79 dans le champ SPN Value
15. Cliquer sur le bouton Set SPN Value
16. Vérifier la saisie automatique de la valeur entrée
17. Cocher l’option Send Once
18. Cliquer sur le bouton ADD
19. Choisir l’onglet ECO FILE
20. Cliquer sur le bouton Delete FILES
21. Revenir à l’onglet CAN FRAME
22. Cliquer sur le bouton Generate
23. Cliquer sur le bouton Oui du POPUP qui s’affiche
24. Attendre 8 secondes
25. Choisir l’onglet ECO FILE
26. Cliquer sur le bouton Import FILES
27. Saisir le chemin « C:/TGU FILES »
28. Cliquer sur Enregistrer
29. Aller au chemin « C:/TGU FILES » et décompresser le fichier « CompressedFolder.tar.gz »
30. Revenir à notre application
31. Cliquer sur Open File
32. Choisir un fichier dans du chemin « C:/TGU FILES/test »
33. Vérifier qu’il s’agit le fichier relatif à l’excès de véhicule

# Conclusion

Nous avons présenté dans ce chapitre les trois derniers sprints, en décrivant leurs back logs et leurs phases d’analyse, ainsi qu’une conception et réalisation du produit final.

# Conclusion générale

Les révolutions technologiques dans l’informatique ont rendu possible le développement des nouveaux services qui fournissent une grande valeur ajoutée par rapport à certains services traditionnels. Dans ce contexte s’inscrit le présent projet de fin d’études qui a comme objectif d’automatiser et améliorer le bon fonctionnement des tests.

Il est devenu évident que la méthode de l'automatisation de test est un processus primordial dans le développement logiciel et le cycle de vie des produits informatiques vu le gain de temps et de coût liés à la réutilisation des scénarios de tests.

Ce rapport décrit le bilan de notre travail, et afin de bien accomplir notre tâche nous avons procédé, en premier lieu à une présentation du cadre générale du projet et du choix méthodologique suivi. Par la suite, nous avons entamé l’analyse et la spécification des besoins en les modélisant par un diagramme de cas d’utilisation général. Aussi, nous avons présenté la conception et l’architecture du projet en citant les différentes technologies utilisées. En dernier lieu, nous avons détaillé la conception de notre solution proposée et la réalisation de chaque sprint.

Sur le plan professionnel, ce stage a été une occasion pour assimiler les différents outils et les connaissances acquises au cours de mes années d'études et bien évidement d’avoir la chance de me familiariser avec les nouvelles technologies qui ont une valeur bien connue dans le marché des technologies.

Cependant, le projet peut être considérablement étendu par l’ajout de nouvelles fonctionnalités qui permettent de le généraliser et de le mettre en service pour tous les projets ARDIA utilisant la communication CAN comme technologie de validation.

# Bibliographie

[1] : **Présentation du GROUPE ACTIA,** <http://www.actia.com> : Consulté le 10/03/2019

[2] : **Présentation d’organisme d’accueil,** <http://www.ardia.com.tn/> : Consulté le 10/03/2019

[3] : **Méthodologie SCRUM,** <https://www.scrum.org/> : Consulté le 06/04/2019

[4] : **Qt Creator,** <https://fr.wikipedia.org/wiki/Qt_Creator>: Consulté le 06/04/2019

[5] : **Ranorex Studio,** <https://economictimes.indiatimes.com/definition/ranorex-tool>: Consulté le 06/04/2019

[6] : **Qt FramWork,** <https://www.techopedia.com/definition/13148/qt>: Consulté le 22/05/2019

[7] : **C++,** <http://www.dicodunet.com/definitions/developpement/c-plus-plus.htm>: Consulté le 22/05/2019

[8] : **C#,** <https://www.techopedia.com/definition/26272/c-sharp>: Consulté le 22/05/2019

# Annexe

1. **Vérification et Validation**

La vérification et la validation, ou encore dit techniquement on dit V&V, présentent deux disciplines de la qualité logicielle. Dans ce cas où ses procédés soient approuvés et appliqués absolument et avec une vive faculté d’entendement et d’intelligence, dès le début du projet, ils peuvent respectivement intensifier la qualité du logiciel. En outre, l’acronyme V&V assure :

* Vérification : est-ce que le logiciel fonctionne correctement ?
* Validation : est-ce que le logiciel fait ce que le client veut ?

En terme qualité logicielle, un système peut s’engraver suite à une fausse exigence, une conception présentant des imperfections ou un code défectueux. Donc, on recourt à la vérification afin de discerner les défauts, les corriger ou les éliminer.

1. **Test boite noir**

On pourvoit à cette méthode où le test se déroule sans avoir aucune connaissance du fonctionnement intérieur de l'application c’est-à-dire sans percevoir son code. Le testeur est inconscient de la conception interne du système et n'a pas accès au code source. Spécifiquement, en effectuant un test de boîte noire, un testeur détient trois aspirations : les données en entrée, l'objet à tester et les observations attendues.

1. **Test boite blanche**

Les tests en « boîte blanche » consistent à vérifier le bon agissement d’une application et son architecture interne, son mécanisme, ses processus, plutôt que ses fonctionnalités. Par l’intermédiaire du code source de l’application, le testeur vise à analyser un programme informatique dont on connait la logique interne.

1. **Test boite grise**

Cette méthode du test demande certaines connaissances sur le mécanisme interne de l’application à tester, c’est-à-dire elle sollicite à la fois ses fonctionnalités et son fonctionnement. Le testeur connaît le rôle du système et de ses propriétés, et a également un acquis, bien que relativement limitée, de ses particularités internes comme la structure des données internes et les algorithmes et les modèles utilisés. Contrairement aux tests boîte noire où le testeur ne vérifie que l'interface du l’application, dans les tests boîte grise le testeur a accès aux documents de conception et à la base de données. Ayant cette connaissance, un testeur peut préparer les meilleurs données et scénarios des tests lorsqu’il vise à établir leurs plans.

1. **Test unitaire**

Ce type de tests est utilisé pour confirmer des structures au niveau du code (méthodes, classes...), il permet au testeur de s'assurer qu'une unité de code n’inclut pas d'erreur de programmation et d’instituer une correction fonctionnelle des modules autonomes. Les tests unitaires sont généralement effectués avant que le code soit remis à l'équipe du test et validation pour exécuter formellement les cas du test. Au cours du test, le développeur des scripts isole chaque partie du programme tout en montrant que les unités de code sont correctes en termes d'exigences et de fonctionnalités. L'objectif principal est d'isoler chaque unité du système pour identifier, analyser et corriger les défauts.

1. **Test fonctionnel**

Le test fonctionnel vérifie que chaque fonction de l'application logicielle fonctionne conformément à la spécification de l'exigence client. Ce test implique principalement des tests de boîtes noires sans se préoccuper du code source de l'application. Toutes les fonctionnalités du système à vérifier sont testées, il fallait fournir une entrée appropriée, vérifier la sortie et ensuite comparer les résultats réels avec les résultats estimés. Ce test consiste à vérifier l'interface utilisateur, les API, la base de données, la sécurité et les applications client/serveur. Lors d’un test fonctionnel, le testeur doit passer par les étapes qui suivent :

* La détermination de la fonctionnalité que l’application, sous test, doit accomplir.
* La détermination des données du test en fonction des spécifications du système à tester.
* Les résultats sont basés sur les données du test et sur les spécifications du système.
* L'écriture de scénarios et du script approprié ainsi que l’exécution des cas du test.
* La comparaison des résultats est basée sur les cas du test exécutés et la spécification.

1. **Test système**

Les tests système de logiciel permettent de vérifier les fonctionnalités du système en son ensemble. Selon l’ISTQB le test système est le processus de test d'un système intégré pour vérifier qu'il répond aux spécifications et aux exigences. Les tests système appartiennent à la classe des tests de type boîte noire puisqu’ils n’exigent aucune connaissance du fonctionnement interne du système ou du code source de l’application sous test. Une fois que tous les composants du système sont intégrés, le testeur recourt test système dans les cas suivants :

* Au cours du cycle de vie du développement logiciel, le test du système est le premier niveau de test qualité.
* Au cours de l’étape de vérification des exigences fonctionnelles.
* Au cours de la validation des exigences de l'architecture d'application.
* Au cours de l’intégration de l'application ou bien du système dans son environnement.

1. **Test d’intégration**

Le test d'intégration, également appelé intégration et test, en acronyme technique I & T, est un processus de développement de logiciels dont les unités de programme sont combinées et testées en tant que groupes de multiples façons. Ces parties combinées permettent de déterminer si le système sous test fonctionne correctement ou non. En effet, il existe deux manières primordiales de réaliser un test d'intégration, appelées méthode ascendante et méthode descendante. Les tests d'intégration ascendante commencent par des tests unitaires, suivis de tests de combinaisons progressivement plus élevées d'unités appelées modules. Cependant, pour les tests d'intégration descendants, il faut tester les modules de haut niveau en premier lieu. Ensuite, on s’adresse progressivement aux modules de niveau inférieur.

1. **Test de régression**

Le test de régression est le processus consistant à tester les modifications apportées aux programmes informatiques afin de s'assurer que l'ancienne programmation fonctionne toujours avec les nouveaux changements subis. Les tests de régression présentent une partie normale du processus de développement du programme réalisé par des spécialistes des tests de code. Les codeurs du département d'essai développent des scénarios de test de code et des pratiques qui testeront de nouvelles unités de code après leur écriture. Avant qu'une nouvelle version d'un produit logiciel ne soit diffusée, les anciens cas de test sont exécutés contre la nouvelle version afin de s'assurer que toutes les anciennes fonctionnalités fonctionnent encore. La raison de dysfonctionnement peut être que modifier ou ajouter de nouveaux bouts de codes au programme source peut facilement introduire des erreurs car le code doit être invariable.

1. **Test d’acceptation**

Le test d’acceptation, ou encore test recette, est une étape de développement des projets, qui vise à assurer intégralement que le produit est conforme aux spécifications client. Ce test est mené par l'équipe d’assurance qualité logicielle qui permet d'évaluer si le système sous test répond à ses spécifications et s’il répond aux exigences du client. Le testeur dispose d’un ensemble de scénarios pré-écrits dans un plan de validation, ainsi que des cas de test, qui sera utilisé pour tester l’application. En effet, les tests d'acceptation ne sont pas destinés seulement, à signaler les erreurs d'orthographe simples ou les lacunes de l'interface, mais d’autre part à signaler les bugs du système qui se traduira par le plantage du système ou ses erreurs majeures.

1. **Test de performance**

Un test de performance est un test dont la finalité est d’évaluer la performance d'un système informatique. On recourt à ce type de test afin de mesurer les temps de réponse d'un système applicatif en fonction de sa stimulation. Ce type de test peut mesurer le comportement d'un système en fonction de la charge d'utilisateurs simultanés ou le nombre des itérations répétées d’un cas de test. Il est considéré comme l'un des tests obligatoires pour tester le programme au niveau des aspects rapidité, temps de réponse, stabilité, etc.

1. **Test de robustesse**

Le test de robustesse permet d’évaluer la puissance d’un système sous test, et spécifiquement son aptitude à contenter certaines propriétés critiques dans un environnement incertain. Son objectif primordial est de vérifier le degré auquel un système informatique peut fonctionner convenablement en présence d'intrants invalides ou bien lors d’une commande de conditions environnementales stressantes. Ce type de test permet d'analyser le système dans le cas où ses capacités sont saturées ou bien d’examiner les réponses du système aux perturbations proche ou hors des limites des domaines de définition des entrées.

1. **Test de validation**

Le test de validation permet de vérifier si toutes les exigences client, décrites dans le document de spécification du logiciel, sont respectées ou non. Pour effectuer un test de validation, le testeur doit passer par les étapes suivantes :

* Validation fonctionnelle : les tests fonctionnels approvisionnent que les différents modules ou composants adaptent correctement les exigences client.
* Validation solution : les tests solution assurent que les exigences client sont considérées d'un point de vue cas d'utilisation de sorte que chaque grand cas d'utilisation est validé séparément, puis tous les cas d'utilisation sont validés ensemble.
* Validation performance et robustesse : les tests de performance vérifient la correspondance de la solution par rapport à ses exigences, tandis que les tests de robustesse permet de mettre en évidence des problèmes de stabilité et de fiabilité

1. **Test de sécurité**

Le test de sécurité, ou encore dit vulnérabilité, consiste à tester un programme afin d'identifier les défauts et les lacunes du point de vue sûreté et endurance. Il permet de découvrir les failles d’un système et de mesurer les conséquences qu’elle peut générer. Un test de sécurité doit également garantir la confidentialité, l’authentification, la disponibilité au cours du temps, et l’autorisation d’un système d’information. Par conséquent, on peut classifier le test logiciel sur trois grands axes :

* Le niveau d’avancement dans le développement.
* Le niveau d'accessibilité.
* La caractéristique c’est-à-dire ce qu’on veut tester.

1. **Test automatique**

En génie logiciel, un test indique une procédure de vérification et de validation partielle d'un système. Son objectif essentiel est d'identifier un nombre optimal de comportements problématiques du système afin d'en améliorer sa qualité et sa fiabilité. Dès son existence, le test a subi un progrès technique assez remarquable en point de vue outils, stratégies, langages et méthodes. Cette grande croissance nous a fait découvrir le concept de l’Automatisation des tests. En effet, l'objectif du test automatisé est d’abréger autant que possible l'effort de test avec un ensemble minimal de scripts. Les outils d’automatisation sont capables d’écrire des scripts avec des cas de tests déterminés, d'exécuter des tests, d’apporter les résultats et de les comparer avec des essais antérieurs. En outre, ces tests automatisés avec ces outils peuvent être exécutés à plusieurs reprises, à tout moment de la journée, avec de nombre d’itérations prédéfinis. La raison primitive pour l’automatisation des tests est le temps. Le testeur a besoin énormément de temps pour vérifier les nouvelles fonctionnalités de l’application sous test. Proportionnellement, plus l’application évolue, plus on a besoin de temps pour vérifier le mécanisme du système et de balayer toutes ses propriétés, ce qui peut accroitre le risque de l’apparition des bugs.