Une image contenant Police, Graphique, logo, symbole

Description générée automatiquementUne image contenant symbole, Graphique, Police, capture d’écran

Description générée automatiquement

DOSSIER DE VALIDATION

Concepteur Développeur d’Application Numérique

Pour le Titre RNCP 36463

|  |  |
| --- | --- |
| Nom Prénom | Corin Gaëtan |
| Nom Prénom du tuteur | Craig Olivier |
| Acronyme de ma certification IPI visée | CDAN |
| Niveau visé | RNCP6 |
| Date de la soutenance | 14 septembre 2023 |
| Lieu de la soutenance | Blagnac |

Une image contenant symbole, Graphique, Police, capture d’écran

Description générée automatiquement

Table des matières

[Tableau des compétences 4](#_Toc141306290)

[Présentation Personnelle 6](#_Toc141306291)

[Présentation de l’entreprise 6](#_Toc141306292)

[Historique 6](#_Toc141306293)

[Activités et Chiffres-clé 7](#_Toc141306294)

[Mon équipe et mes activités 7](#_Toc141306295)

[Présentation du client 8](#_Toc141306296)

[Équipe et activités 8](#_Toc141306297)

[Besoins exprimés 9](#_Toc141306298)

[Définition du projet 10](#_Toc141306299)

[Cahier des charges 10](#_Toc141306300)

[Architecture des interfaces 11](#_Toc141306301)

[Architecture logicielle 12](#_Toc141306302)

[Back-end 12](#_Toc141306303)

[Front-end 13](#_Toc141306304)

[Méthodologie du projet 14](#_Toc141306305)

[Planning et Livraisons 14](#_Toc141306306)

[Logiciels de travail collaboratif 14](#_Toc141306307)

[Méthode Agile 15](#_Toc141306308)

[Réunions client 15](#_Toc141306309)

[Implémentation du composant log et de son service 16](#_Toc141306310)

[Contexte et objectif 16](#_Toc141306311)

[Interface du composant 16](#_Toc141306312)

[Mise en œuvre et réalisation 17](#_Toc141306313)

[Récupération de la donnée 17](#_Toc141306314)

[Architecture du service 18](#_Toc141306315)

[Architecture du composant 19](#_Toc141306316)

[Problématique rencontrée et solution trouvée 19](#_Toc141306317)

[Identification de la problématique 19](#_Toc141306318)

[Réunion client 19](#_Toc141306319)

[Nouvelle solution implémentée 19](#_Toc141306320)

[Transformation du projet en livrable 19](#_Toc141306321)

[Création d’un Dockerfile 19](#_Toc141306322)

[Création d’une documentation technique 19](#_Toc141306323)

[Validation du livrable final par le client 19](#_Toc141306324)

[Projet personnel : Outil de sauvegarde et de suivis des analyses d’eau 19](#_Toc141306325)

[Présentation du projet 19](#_Toc141306326)

[Cahier des charges 20](#_Toc141306327)

[Conception UML 20](#_Toc141306328)

[Conception MCD et MLD 20](#_Toc141306329)

[Maquettage 20](#_Toc141306330)

[Architecture logicielle 20](#_Toc141306331)

[Tests Unitaires 20](#_Toc141306332)

[Déploiement sur serveur 20](#_Toc141306333)

[Conclusion 20](#_Toc141306334)

[Bibliographie et citations 20](#_Toc141306335)

[Glossaire 20](#_Toc141306336)

[Table des illustrations 21](#_Toc141306337)

[Annexes 21](#_Toc141306338)

# Tableau des compétences

|  |  |
| --- | --- |
|  | Pages correspondantes |
| BC01 : Concevoir des applications numériques en intégrant les recommandations de sécurité | |
| Formaliser les procédures des services utilisateurs pour recenser les résultats attendus. |  |
| Prendre en compte les impératifs utilisateurs en respectant les contraintes des recommandations qualité de la norme en vigueur pour l’architecture des logiciels. | Définition du projet :cahier des charges |
| Concevoir l’architecture d’une solution fiable en identifiant les spécificités d’une activité pour produire du logiciel générique réutilisable. | Implémentation du composant log et de son service:Contexte et objectif |
| Concevoir des services d’accès aux données indépendants du mode de stockage en garantissant la sécurité des données pour produire du logiciel partageable. |  |
| Envisager toutes les possibilités, même les plus improbables pour livrer un logiciel déterminé en recherchant systématiquement l’erreur ou le dysfonctionnement. | Problématique rencontrée et solution trouvée:Identification de la problématique |
| Estimer la charge de traitement et la puissance de calculs nécessaire proportionnellement aux nombres d’utilisateurs simultanés en vue d’anticiper les évolutions. |  |
| Respecter une norme de présentation des écrans et documents de sortie en utilisant les outils de maquettage appropriés en vue de permettre l’adaptabilité des sorties garantissant leur l’accessibilité à différents niveaux de handicap. |  |
| Identifier les risques et leur niveau de criticité pour permettre leur prévention. |  |
| Orienter son style de programmation en vue de produire du code lisible, maintenable, robuste, fiable, efficace par une approche méthodologique objet. | Mise en œuvre et réalisation:Architecture du service  Mise en œuvre et réalisation:Architecture du composant |
| Garantir un accès sécurisé aux données en évitant toute corruption de la base de données, par l’usage de contraintes d’intégrité et de déclencheurs. |  |
| BC02 - Piloter un projet DevOps de développement d’application numérique. | |
| Formaliser les procédures des services utilisateurs en contrôlant le respect du management des processus de l’entreprise. |  |
| Réaliser une réingénierie d’un processus de l’entreprise en tenant compte des règles de management de l’entreprise dans un but d’amélioration des résultats et/ou des conditions de travail. | Mon équipe et mes activités |
| Formaliser la circulation des documents générés en identifiant les acteurs concernés et leur rôle ainsi que les rubriques utilisées et leur provenance. | Définition du projet :cahier des charges |
| Modéliser une base de données adaptée aux attentes en formalisant les règles de gestion et d’organisation de l’entreprise des processus concernés et en tenant compte d’un existant possible. |  |
| S’insérer dans l’urbanisation présente et future du système d’informations en concevant des éléments logiciels réutilisables et structurés en couches. | Architecture logiciel :frontend |
| Choisir le degré de réutilisation à utiliser selon le cas de figure en décidant collectivement en équipe d’une réutilisation totale, partielle ou une écriture neuve et des briques à réutiliser en tentant de se rapprocher du niveau maximum de satisfaction CMMI. | Implémentation du composant log et de son service:Contexte et objectif |
| Utiliser l’expérience vécue pour anticiper l’avenir en vue d’estimer des délais de réalisation compte tenu du taux réel de disponibilité du réalisateur et des contraintes date départ/date livraison. | Méthodologie du projet :planning et livraison |
| Coordonner un projet de développement en utilisant les outils et méthodologies de gestion de projet Agile afin de respecter les contraintes définies (coûts, délais, qualité), tout en minimisant les risques. | Méthodologie du projet :méthode agile |
| Clôturer une mission de développement en faisant valider le livrable par les parties concernées et en respectant les préconisations CFTL. |  |
| Adapter son discours à l’auditoire en appuyant ses manipulations et explications fonctionnelles à partir des contraintes exprimées tout au long du projet pour obtenir une bonne adhésion des décideurs ou de leurs représentants. | Méthodologie du projet :réunions client  Problématique rencontrée et solution trouvée:Réunion client |
| Réaliser la procédure d’intégrabilité d’un logiciel ou d’un correctif dans l’environnement de tests selon les bonnes pratiques ITIL en vérifiant que l’intégralité des points de contrôles sont positifs. |  |
| Interagir efficacement dans un environnement de travail collaboratif en reformulant la demande et en adaptant son discours à l’auditoire pour obtenir un niveau de compréhension des demandes optimum en tenant compte des collaborateurs en situation de handicap. | Problématique rencontrée et solution trouvée:Réunion client |
| User d’une communication professionnelle tant en français qu’en anglais en structurant des informations sur une thématique donnée afin de les partager au sein de la structure ou à l’externe. | Méthodologie du projet :méthode agile Méthodologie du projet : réunions client |
| BC03 - Développer des applications numériques. | |
| Utiliser les ressources à sa disposition ou faire appel si besoin à un expert externe pour contrôler l’identification et la teneur du résultat attendu ou approfondir un point technique. | Problématique rencontrée et solution trouvée:Réunion client |
| Décomposer un problème complexe en sous-problèmes en faisant des analogies et des différenciations tout en changeant d’approche, de point de vue, face à un obstacle en vue de résoudre un problème algorithmique. | Mise en œuvre et réalisation:Architecture du service  Mise en œuvre et réalisation:Architecture du composant |
| Traduire une solution algorithmique dans un langage de codage informatique avec l’utilitaire approprié. |  |
| Modifier un algorithme sans générer de dysfonctionnements en comprenant et s’adaptant si besoin au mode de pensée de son auteur. |  |
| Remédier aux erreurs de codage ou de logique en comprenant ou interprétant les messages d’erreur du compilateur ou du système d’exploitation pour mettre au point un élément logiciel opérationnel. |  |
| Intégrer des éléments logiciels hétérogènes spécifiques et en réutilisant des services logiciels externes, en local ou à distance pour produire des exécutables livrables en conformité avec la politique RSE. | Architecture logiciel :frontend |
| Préparer des jeux d’essai en envisageant toutes les possibilités dans le but de livrer un logiciel déterminé exempt d’anomalies logiques et fonctionnelles. |  |
| Estimer son taux de disponibilité réel et rendre compte de son travail en renseignant l’outil de suivi permettant de constater l’avance de la tâche en cours et sa répercussion sur l’ensemble du projet. | Méthodologie du projet :planning et livraison  Méthodologie du projet :logiciel de travail collaboratif |
| BC04 - Réaliser une interface d’échange de données informatisées. | |
| Procéder à une analyse organique d’un logiciel existant par l’étude du code des programmes et des données qui sont accédées dans le but de disposer d’une documentation technique du logiciel jusque-là inexistante ou indisponible. |  |
| Faire des analogies et des différenciations entre les données à échanger entre logiciels à l’aide des dictionnaires de données disponibles ou reconstitués en vue de permettre l’échange de données entre les logiciels identifiés. | Architecture logiciel :frontend |
| Produire des données indisponibles en agrégeant, consolidant ou calculant automatiquement ces nouvelles données à partir de celles existantes dans le but de favoriser les échanges de données entre logiciels dans le respect de la RGPD. |  |
| Permettre l’exportation et l’importation de données entre logiciels en utilisant des formats compatibles entre les systèmes émetteurs et récepteurs grâce à des flux synchrones ou asynchrones. | Architecture logiciel :backend Mise en œuvre et réalisation:Architecture du service |
| Écrire des scripts système en langage de commande ou en shell système pour automatiser l’installation, la configuration de systèmes d’exploitation et de middleware permettant la création, la configuration de machines virtuelles, de serveurs d’applications, Web et bases de données dans le but d’adapter et simuler en réel l’environnement d’exécution du logiciel à tester. |  |

# Présentation Personnelle

Mon nom est Gaëtan Corin et j’ai 28 ans. Il y a de cela 2 ans, j’ai entamé une reconversion dans le domaine du développement web et logiciel.  
  
Mon parcours initial est bien différent. J’ai commencé mon parcours professionnel dès l’âge de 15 ans en réalisant un CAP boulanger en alternance, suivis d’un CAP pâtissier.

J’ai eu la chance de pouvoir travailler dans de nombreuses boulangeries pâtisseries artisanales sur Toulouse. Étant de nature curieux, j’ai rapidement cherché à découvrir les différentes manières de procéder, manière de faire, dans ce monde ou la baguette est roi.

Durant l’année 2019, je me suis lancé dans le monde de l’entreprenariat en plus de mon travail de boulanger. Je me suis déclaré Auto-entrepreneur, puis j’ai commencé a réalisé différentes missions pour des clients diverses.   
  
C’est durant cette période que j’ai réalisé mon premier site web en NoCode en utilisant Shopify. J’ai pu découvrir avec fascination l’univers de la création Web jusqu’à décider d’apprendre à coder sur mon temps libre mes premières pages Web en suivant des didacticiels.   
C’est aussi durant cette période que j’ai eu la chance de discuter avec un voisin de résidence, tout juste jeune diplômé Ingénieur Web, qui m’a expliqué plus en détail sa profession ainsi que la journée type d’un développeur professionnel.  
  
J’ai donc naturellement pris l’initiative de m’inscrire en septembre 2021 au centre de l’Adrar pôle numérique pour une formation intensive de 11 mois dans le développement web et mobile.

A la suite de cela, j’ai souhaité poursuivre mes études une année de plus au sein du campus IPI pour suivre une formation de Concepteur Développeur d’Application Numérique. J’ai également pu intégrer les équipes de CAT, faisant partie de l’ESN CELAD, afin de réaliser mon alternance.

# Présentation de l’entreprise

## Historique

J'ai effectué l’intégralité de mon alternance au sein de l'entreprise CELAD à Toulouse. Il s’agit d’une Entreprise de Service Numérique (ESN).

Créée à Toulouse en 1990, elle s’est spécialisée principalement dans la maîtrise des technologies du domaine de l’informatique industrielle et de l’ingénierie des systèmes d’informations. L’entreprise ne se restreint donc pas qu’à l’informatique pure mais propose également son expertise dans le domaine de l’architecture ou des systèmes centralisés.

Au fil des années, l'entreprise a étendu sa présence dans de nombreuses villes Française.  
Elle s’est implantée à Paris en 2003, Lyon en 2006, Nantes en 2007, Aix en 2010, Rennes en 2011, Nice en 2012, Bordeaux en 2016 et Strasbourg en 2017.

## Activités et Chiffres-clé

Une image contenant texte, capture d’écran, diagramme, ligne

Description générée automatiquementDepuis sa création, le chiffres d’affaires ainsi que le nombre de collaborateurs de CELAD ne cesse d’évoluer.

Figure 1 Chiffre d’affaires et collaborateurs

Le chiffre d’affaires s’élevait en 2022 à 125 millions d’euros, ce qui représente une augmentation de 25% par rapport à 2021. Le nombre de collaborateurs, quant à lui, est passé de 1300 à 1500.

La véritable richesse de cette entreprise repose sur les compétences de ces collaborateurs.  
En effet, au cours des années, CELAD a développé une véritable expertise dans les domaines Bancaire et Assurance, le Transport, la Santé, Aéro et Spatial, le Commerce et la Distribution, le Télécom, le Numérique, l’Énergie ainsi que l’Administration.

Elle travaille aussi bien avec des grands groupes que des Petites et Moyennes Entreprises. [1]

## Mon équipe et mes activités

J’ai réalisé mon année d’alternance dans une équipe interne à CELAD au nom de CAT.

CAT signifie « Celad Automation Tools ». Il s’agit d’un Framework de tests développé en interne qui permet d’effectuer de la validation fonctionnelle complète et des tests système.  
Ce Framework est accompagné d’une Application Web, permettant d’avoir une interface afin d’afficher les résultats des tests de CAT.

L’Equipe se compose de 8 développeurs, dont mon chef de projet qui est aussi le Product owner, et une développeuse qui est aussi le Scrum master.

Cette équipe travaille en méthode Agile, en réalisant des sprints Planning de 15 jours. Cela nous permet de définir des tickets Jira pour chaque développeur représentant notre travail à réaliser pour le sprint. Une fois le temps imparti, nous réalisons ensuite un retro-planning afin de faire un bilan du sprint précédent.  
  
À la suite de ma demande, nous avons mis en place un Daily durant mon alternance. Le Daily est une réunion quotidienne afin de pouvoir suivre l’avancer et les points de blocage de chaque collaborateur.  
   
Durant cet année, j’ai principalement travaillé sur l’application web de CAT, puis j’ai ensuite travaillé pour le client Renault au compte de l’équipe CAT.

# Présentation du client

## Équipe et activités

En début d’année, notre équipe a été démarché par l’équipe FUTE Renault.  
Il s’agit d’une équipe de collaborateurs interne à l’ESN CELAD qui travaille exclusivement en sous-traitance pour le compte de Renault.

L’équipe FUTE a développé et mise en place un logiciel spécialisé pour réaliser des tests lors de la mise à jour de véhicules.

Une image contenant texte, capture d’écran, Police, logo

Description générée automatiquementEn effet, depuis l’arrivée de l’informatique dans les voitures durant ces dix dernières années, il est nécessaire de réaliser des mises à jour récurrentes sur les véhicules en fonctionnement afin d’améliorer la sécurité, corriger les éventuels bugs ou bien intégrer de nouvelle fonctionnalité.  
  
Afin de pouvoir mettre à jour ces véhicules, le logiciel FUTE a été créé. Il est en lien direct avec le TECU de la voiture.  
  
TECU signifie « Transmission Electronic Control Unit ». Il s’agit du module électronique interne à la voiture en charge de capter et collecter l’intégralité des informations du véhicule. C’est ce module qui définit l’état du véhicule servant à la base à optimiser les changements de vitesse et les performances globales de transmission.  
  
En ce connectant au TECU de la voiture, le logiciel FUTE peut donc accéder à l’intégralité des informations du véhicule.  
Ainsi, il peut réaliser des campagnes de tests, permettant de simuler des nouvelles mises à jour.

C’est grâce aux résultats de ces tests que l’équipe FUTE peut déceler différentes bugs ou correctifs à améliorer, avant de mettre en production la mise à jour sur l’ensemble du parc automobile.

Figure 2 Interaction entre le TECU et le logiciel FUTE

## Besoins exprimés

Le logiciel FUTE génère ses résultats de campagnes de tests sous format de fichier texte, rangés dans un dossier unique externe au logiciel. Ces fichiers contiennent l’intégralité des lignes de logs générés par FUTE, et c’est en analysant ces fichiers que les utilisateurs obtiennent les résultats des tests.

A la suite de retours d’utilisateurs interne à Renault, il a été remonté une difficulté persistante à lire et analyser ces fichiers logs générés par le logiciel.

En effet, la génération de logs est très importante, et le fait d’avoir des fichiers contenant plusieurs centaines de milliers de lignes de logs est difficilement compréhensible pour un utilisateur souhaitant analyser la réaction d’un test sur un véhicule.

De plus, l’état globale du véhicule doit pouvoir être analysé afin que l’utilisateur puisse comprendre les réactions sous-jacentes du véhicule lors des tests de mises à jour. Pourtant, de nombreux paramètres sont définis ou initiés sur des tests précédents, ce qui rend l’état du véhicule incompréhensible car les informations importantes peuvent être séparés par plusieurs dizaines de milliers de lignes de logs les uns des autres.

Suite à ces problématiques identifiées, l'équipe FUTE a fait appel à nos services, afin que nous leurs développions une application Web, permettant de lire les fichiers logs générés par le logiciel FUTE et d'afficher de manière dynamique l'état global du véhicule lors de l'évolution de chaque test d'une campagne.

Le choix de nous confier cette tâche s’est avéré stratégiquement intéressant car le logiciel FUTE présente de nombreuses similitudes avec notre Framework de Testing CAT que nous développons au sein de notre équipe. De plus, notre expérience prouvée avec notre application Web destinée à afficher les résultats des tests de CAT a été un critère déterminant pour l'obtention de ce contrat.

Ayant eu la chance de monter en compétences pendant plus de six mois sur l’Application Web relié à notre Framework, mon chef de projet m’a donné l’opportunité de réaliser le développement de cette nouvelle application en autonomie et sous sa responsabilité.

# Définition du projet

## Cahier des charges

Dans le but de définir de manière plus précise les attentes du client, nous avons élaboré un cahier des charges. Nous y avons défini les différents acteurs ainsi que les rôles de chacun afin que l’équipe FUTE puisse utiliser ce document pour valider le montant de la facturation du projet.

Durant nos interactions avec le client, les attentes ont été clairement exprimés, ce qui nous a grandement aidé dans la compréhension et le résultat attendu du projet.  
  
L’objectif principal de cette application consiste à récupérer les données d’un fichier log contenu dans le dossier partagé FUTE, afin d’en extraire les informations les plus pertinentes qui seront ensuite affiché sur l’application Web.  
Les informations dite « pertinente » auront au préalablement été marquée au sein des lignes de logs par les caractères « GUI », qui signifie Graphical User Interface. Elles auront été reformatées et nommé par type afin de faciliter le parsing des informations.

Ces informations devront ensuite être triés par l’application et afficher par type de manière dynamique. L’objectif étant de pouvoir voir à un instant donné l’état global du véhicule.

Figure 3 Exemple de ligne de log GUI de type signal update

L’ensemble des lignes de logs sera aussi affiché par l’application et segmenté par test, afin que l’utilisateur puisse avoir la possibilité de disposer d’un maximum d’informations directement sur l’application sans jamais avoir à ouvrir le fichier de log.  
  
Afin que l’utilisateur puisse naviguer entre les tests, ainsi que à l’intérieur de chaque test, une barre de défilement de temps identique à celle d’une vidéo permettra de se déplacer dans la temporalité, et de rejouer les différents tests de la même manière qu’ils ont été exécutés.

L’application devra avoir la capacité de lire un fichier de log qui est en train d’être édité par le logiciel FUTE, permettant d’avoir un aperçu en temps réel des résultats de la campagne de test en train d’être exécuté.  
  
Enfin, un listing des fichiers de log permettra de sélectionner et rejouer les différentes campagnes de tests. Ce listing devra être mis à jour automatiquement à chaque création d’un nouveau fichier par le logiciel FUTE.  
  
L’application devra être accessible, performant et sécurisé afin de respecter les contraintes et recommandations qualité de la norme Renault. Une attention particulière a été fait sur la norme WCAG (Web Content Accessibility Guidelines), qui implique de concevoir une interface utilisateur qui peut être utilisée par des personnes ayant des limitations visuelles.

## Une image contenant texte, capture d’écran, Police, ligne Description générée automatiquementArchitecture des interfaces

Figure 4 Parcours utilisateur de l'application

L’interface utilisateur se compose de deux pages.  
La première, appelé « Home », est la page qui va lister l’intégralité des fichiers logs disponible afin que l’utilisateur puisse choisir la campagne de tests qu’il souhaite rejouer.  
La seconde page, appelé « Test Result », apparait une fois la campagne choisie et se compose de trois interfaces : Test, State et Signal.

Une image contenant texte, capture d’écran, Police, conception

Description générée automatiquement

Ces trois interfaces sont destinées à afficher l’état global du véhicule en affichant toutes les données GUI rangés par type. Voici un exemple de composant affichant des données GUI de type « vehicle state ». Il s’agit ici de donnée représentant l’état du contact de la clé du véhicule, ainsi que son état précédent.

Une image contenant texte, Police, capture d’écran, nombre

Description générée automatiquementToutes ces interfaces ont deux composants en commun.  
Le premier est la barre de défilement du temps. C’est l’outil qui permet à l’utilisateur de choisir le test et le temps précis sur lequel il souhaite afficher l’état du véhicule. Les données GUI vont par conséquent apparaître ou disparaître afin d’afficher l’état exacte du véhicule lors du temps sélectionné. On pourra donc avoir une visualisation complète de l’état global du véhicule à chaque étape d’un test.  
  
  
Le second composant affiché sur toutes les interfaces est l’affichage des lignes de logs pour le test qui été en cours d’exécution au moment du temps sélectionné avec la barre de défilement.

Figure 5 Composant d'affichage des logs

Figure 6 Barre de défilement du temps

Des représentations plus détaillées de ces pages et interfaces sont disponibles dans l’annexe.

## Architecture logicielle

### Back-end

Afin de réaliser cette Application Web, j’ai privilégié le langage Python pour la partie Back-end du projet.  
En effet, l’application nécessite que je développe un outil permettant de répertorier les différents fichiers logs dans le dossier partagé. Il nécessite aussi que je développe un second outil afin d’ouvrir le fichier sélectionné et parser les lignes de logs. Cet outil détectera aussi les changements lors d’éditions de nouvelles lignes de logs par le logiciel FUTE.

Python offre une multitude de bibliothèques et des modules puissants qui facilitent le développement de ces outils, permettant d’unifier le langage entre la gestion serveur du Back-end et ces outils spécifiques.

Pour la création du serveur, j’ai choisi le micro-Framework Flask afin de gérer les connections API et Websocket. Il permet une flexibilité et une simplicité de développement, idéal pour des application web légère. De plus, il permet une intégration de grande variété de bibliothèques Python, permettant au projet de prendre de l’envergure suivant les besoins futurs de l’équipe FUTE.

Voici un exemple de fonctionnement de l’architecture logicielle Back-end lors de la lecture d’une ligne log GUI afin de créer un Object « Event » qui pourra être fournis au Front-end.

Une image contenant texte, capture d’écran, ligne, diagramme

Description générée automatiquementEn premier, l’outil Logparser parse la ligne de log.  
Cette outil fonctionne dans un fil (Thread) afin de pouvoir continuer à recevoir des informations du Websocket durant le processus.

A la suite de ce parsing, il est possible de définir le type de cette ligne, signalé ici en « signal update », en lisant le cinquième élément de la ligne de log.

Suivant le type de GUI, différentes valeurs pertinentes sont attribué à un objet selon un principe de clé-valeur. C’est le client qui définit les valeurs dites pertinentes afin de correspondre aux mieux au cas d’utilisations de l’application.  
Ces relation clés-valeurs sont ensuite contenu dans la clé « «data » de l’Event.  
  
Le valeur du « séquence number » est ensuite généré en réalisation le compte du nombre de ligne GUI afin de connaître leur ordre d’apparition.  
  
Une fois que cet Event est généré, il est transmis au Websocket en utilisant un mécanisme de rappel (callback). C'est ce rappel qui déclenche l'envoi de l'Event vers le Front-end en utilisant la méthode "send".

Figure 7 Architecture logicielle Back-end

### Front-end

Pour la partie Front-end de cette application, j’ai choisi le Framework Angular afin de répondre au mieux aux besoins du projet. C’est un Framework Typescript qui permet une structure solide et modulaire de la partie Front-end.  
Son architecture repose sur des services et des composants, ce qui permet une réutilisabilité simplifiée des différents éléments de l’interface. Cette réutilisabilité est indispensable pour le développement de cette application, car certains éléments apparaissent trois fois entre les différentes interfaces. Angular propose aussi des fonctionnalités intégrées pour gérer les API Rest et les Websockets, permettant un envoi d’informations sécurisé avec le Back-end.

Une image contenant texte, capture d’écran, diagramme, Police

Description générée automatiquementVoici un exemple de fonctionnement de l’architecture logicielle Front-end lors de la réception de l’Event au préalablement envoyé par le Back-end Python.  
  
L’Event est en premier réceptionné par le canal de communication Websocket.

L’information est ensuite envoyée vers son service approprié suivant la valeur du type du GUI.  
L’Event est donc envoyé vers le service en charge de l’affichage des signal update, car dans notre exemple, il s’agit d’un type « signal update ».  
 L’information est ensuite traitée et stockée dans le service en que « state », soit un état.

Figure 8 Architecture Front-end du stockage des Events

Un état correspond à l’ensemble de toutes les informations d’un type à un instant donnée. Le nouvel état qui vient d’être créé aura donc toutes les valeurs des précédentes signal update reçus, ainsi que la nouvelle valeur.   
Cet état est ensuite stocké dans la variable de liste des états.

Une image contenant texte, capture d’écran, diagramme, Police

Description générée automatiquement  
L’utilisateur, lorsqu’il souhaitera afficher un état global du véhicule à un instant précis, devra utiliser le composant de barre de défilement du temps appelé ici « Progress bar ».  
La valeur que l’utilisateur aura sélectionné avec cette barre sera ensuite partagé au service « Progress bar Service ».   
L’objectif de ce service est de partagée la valeur à tous les autres services contenant des états par un principe d’abonnement, afin que ces services fournissent à leur composant l’état approprié correspondant aux informations du véhicule à l’instant précis.

Figure 9 Architecture Front-end de l'affichage d'un Event

Cette architecture de stockage des données permet de ne pas devoir récupérer les données dans le fichier log à chaque sélection un instant précis à afficher. De plus, il permet de ne pas recalculer un état pour chaque composant, ce qui améliore la vitesse d’utilisation, mais contribue aussi à la politique RSE (responsabilité sociétale des entreprise) de Renault en minimisant la puissance de calcul de l’application.

# Méthodologie du projet

## Planning et Livraisons

Une image contenant texte, capture d’écran, nombre, ligne

Description générée automatiquementAvec l’aide de mon responsable, nous avons réalisé un planning dans le but de planifier la réalisation de l’application, mais aussi afin de budgétiser le développement de celle-ci.   
  
Le fait de réaliser une planification nous permet de faire une estimation du temps de travail de chaque implémentation, tout en gardant un aperçu de l’avancement du projet lors du développement.

Figure 10 Planning de la réalisation du développement de l'application

C’est durant cette planification que nous avons définis les différentes tâches.  
   
La budgétisation de cette application étant facturé à la tâche, il est donc primordial de convenablement estimer les temps de travail, et de tenir les délais, afin de ne pas être hors budget.  
  
C’est à la fin de chaque tâche que j’ai dû livrer le code directement sur le répertoire Gitlab de Renault, faisant preuve de travail exécuté déclenchant la facturation.

## Logiciels de travail collaboratif

Pour permettre à l'équipe FUTE de suivre l'avancement du projet, d'obtenir des informations sur les tâches en cours de développement ainsi que de mieux organiser le déroulement du projet, nous avons mis en place des tickets sur le logiciel Jira.  
  
Jira est un logiciel de gestion de projet. Il est principalement utilisé par les équipes de développement de logiciels, les équipe de gestion de projet ou bien pour le support client.  
Il permet de suivre, gérer et planifier les tâches, les bugs ainsi que les projets.

Lors de mon développement sur l’application, c’est moi qui suis en charge de la création, du suivi et de la clôture de mes tickets.   
  
Afin de rester en contact avec l’équipe FUTE, j’ai aussi utilisé le logiciel Microsoft Teams.  
Cela m’a permis de faciliter la collaboration et la communication avec eux lors de problèmes techniques demandant leur intervention.

## Méthode Agile

Durant tout mon processus de développement, moi et mon responsable avons mis en place une méthode Agile permettant de toujours rester en contact avec l’équipe FUTE.

Pour cela, la plateforme de gestion de dépôt Gitlab m’a permis de faire relire mon code par mon responsable, puis par un responsable interne à l’équipe FUTE, favorisant ainsi une approche collaborative et itérative du développement.

L’envoi fréquent sur les différents tickets Jira de captures d’écran et de vidéos de l’application présentant l’avancement du projet ont permis d’avoir un retour constant de client, permettant ainsi de consolidé cette méthode agile.

De plus, le travail en méthode Scrum condensé en sprint de 15 jours, ainsi que les temps de Dailys ou j’ai pu expliquer mon avancer ainsi que les points de blocages à mon responsable ont pu grandement améliorer le développement de cette application.

## Réunions client

Afin d’avoir des retours directs avec les clients, mais aussi pour présenter plus en détail l’avancé du projet, trois réunions clients ont été planifié.  
  
La première, en début de projet, a eu pour but de définir les attentes du client, ainsi que la présentation du cahier des charges que nous avons conçu dans le but de faire valider la conception de celui-ci.

La seconde, au milieu du projet, a été planifier après une livraison de l’application sous format Docker. L’objectif a été de récupérer le maximum de retour client, afin de réaliser des premières modifications suivant les retours clients.

La dernière réunion a eu pour but que je présente l’application finalisé à l’intégralité de l’équipe Renault. C’est la validation de cette présentation et du résultat final attendu qui défini la fin de nos services pour cette facturation.  
  
Enfin, des réunions clients non planifié ont eu lieu lors de points de blocage important durant le développement nécessitant leur intervention.

# Implémentation du composant log et de son service

## Contexte et objectif

Lorsque l’utilisateur va utiliser l’application dans le but de rejouer une campagne de tests, les informations les plus pertinentes sont afficher de manière dynamique afin d’avoir l’état du véhicule à l’instant donnée suivant la barre de défilement du temps.  
Cependant, dans des cas d’utilisation bien spécifique, il est nécessaire d’avoir un aperçu des lignes de logs afin de chercher des données bien précises. C’est le but que doit remplir le composant log, appelé « Log Component ».  
  
L’objectif de ce composant est donc d’afficher les lignes de logs segmentés par test en cours, dans le but de faciliter la lecture pour l’utilisateur tout en ne perdant aucune information.  
Le fait de rendre cette information disponible directement sur l’application permet donc à l’utilisateur d’éviter d’ouvrir des fichiers logs qui font souvent plusieurs centaines de milliers de ligne.  
Ce composant est disponible sur les trois interfaces de la page « Test Result », ce qui permet d’avoir un aperçu du composant à n’importe quel endroit ou l’utilisateur est situé dans la navigation des interfaces.  
  
Son service, le « Log Service », va quant à lui contenir l’intégralité des informations du fichier log. Ces données seront transmises lors du parsing du fichier de manière efficiente afin de minimiser au mieux le temps de chargement lorsque l’utilisateur réalisera un changement d’affichage avec la barre de défilement.

## Interface du composant

Une image contenant texte, Police, capture d’écran, nombre

Description générée automatiquementL’interface de ce composant a été conçu de manière la plus simple possible, afin d’éviter les potentielles latence causée par un grand nombre de lignes de log à afficher.

Figure 11 Composant d'affichage des logs

Il consiste en une partie scrollable, permettant d’accéder à l’intégralité des informations sans prendre trop de place sur les interfaces utilisateurs.  
Le formatage des lignes de logs est conçu de la manière le plus semblable à ceux réalisé dans les fichiers logs déjà existants, afin de facilité la lecture et gardé les habitude des utilisateurs.  
Le retour à la ligne entre chaque ligne de log est aussi conservé afin d’améliorer la visibilité de l’information.

## Mise en œuvre et réalisation

### Récupération de la donnée

C’est dans la partie Back-end, lors du parsing de chaque ligne de log que la récupération de la donnée est effectuée pour ce composant.

Afin de réaliser ce parsing, l’outil « LogParser » est utilisé. C’est lui qui va récupérer le nom du fichier à lire, qui va traiter les informations contenues dans chaque ligne de log du fichier sélectionné pour transformer l’information en typage « objet » transférable par protocole de communication Websocket.  
  
Dans le contexte de l’envois de l’intégralité des lignes de log nécessaire à l’affichage du composant « Log Composant », l’outil LogParser va récupérer la ligne de log en cours d’itération par l’outil, puis il va la parser, c’est-à-dire la séparer, en transformant la ligne de typage « string » en un nouveau format de typage « tableau ». Ce tableau va contenir une liste de valeurs contenant chaque mot. Le caractère espace ou tabulation servira de déclenchement entre chaque séparation.

Une fois les mots séparés, la ligne de log est reconstituée en format string avec uniquement un caractère espace entre chaque mot, dans le but de retirer les tabulations qui auraient rendus la compréhension de l’information difficile dans un composant ou la place est restreinte. Afin de réaliser cette exécution, la méthode « remove\_extra\_space » est employé.  
Enfin, cette ligne de log en typage « string » est insérée dans une variable de typage tableau qui contient les autres lignes de logs d’un seul test de la campagne.  
  
Afin d’envoyer les informations au front-end lorsque les lignes de logs passent à un test suivant, l’équipe Renault a créé un GUI de type « test status » et de valeur « Running ». Il contient le nom du nouveau test en cours et permet de segmenter les tests entre eux.

Figure 12 Ligne de log GUI signifiant le déclenchement d'un nouveau test

Il suffit au LogParser d’attendre que cette ligne de log apparaisse pour que celui-ci envoye la totalité du tableau des lignes de logs au Front-end sous format d’un objet. Lors de la création de cet objet, une clé « séquence number » de typage nombre contiendra le décompte des envois afin de connaître la chronologie des logs. L’ensemble des logs, avant d’être envoyé, sont quant a eux formaté comme un bloc de texte de type string afin d’afficher l’ensemble des logs en une fois, en évitant les itérations inutiles sur le composant. Afin de conserver les retour à la ligne, le caractère « \n » est inséré entre chaque ligne lors de la création du bloc.

L’envoi de cet object, appelé désormais un Event, est réalisé par un callback de l’outil Logparser qui déclenche une méthode « send » du Websocket.

En effet, l’outil Logparser fonctionne à l’intérieur d’un thread, sois un fil. L’objectif de ce thread est de pouvoir recevoir en simultané un signal du Websocket qui puisse arrêter le parsing d’un fichier, même si le LogParser est en cours de fonctionnement. Étant donné que le langage Python fonctionne nativement de manière synchrone, il est donc nécessaire d’utiliser un thread afin de traiter de manière parallèle des informations simultanés.

Si la variable contenant les ligne de logs devient trop importante causé par une quantité de lignes trop importantes à l’intérieur d’un même test, le Websocket de la partie Font-end crée un temps de latence sur toute l’application durant le temps de réception de l’information.  
  
Afin d’améliorer l’expérience utilisateur, un seuil de déclenchement maximal d’un envoi d’Event tous les dix milles lignes de log est mis en place. Une variable booléenne est insérée dans l’objet lors de l’envois, permettant de fournir l’information au front-end s’il s’agit d’une réception de bloc de logs d’un nouveau test, ou la suite du même test précédemment reçu.

### Architecture du service

Sur la partie Front-end de l’application, c’est le service « Fute Event Service » qui est à la charge du Websocket. C’est ce service qui réceptionne les Events envoyé par le Back-end.

A la réception d’un Event, le service lit la valeur de la clé « type » de l’objet, afin de connaître le contenu de l’information et le transmet à son service approprié. Cette clé est définie par une valeur de chaîne de caractère « log » lorsqu’il s’agit de l’envoi de la liste de logs.

Dans ce cas-là, l’information est donc transmise au Log Service.  
  
La réception de l’information par le Log Service se fait par la méthode « onEvent ».

Il s’agit d’une convention de nommage que j’ai définie à la création du projet. En effet, tous les services de la partie Front-end ont une méthode « onEvent » pour réceptionner les informations. Cette convention de nommage permet une meilleure lecture du code.

Dans le Log Service, la méthode « onEvent » reçoit et traite l’information reçu en argument.  
Cette méthode vérifie en premier la valeur de la clé de l’Event « is new test ».  
S’il s’agit d’un nouveau test, le Log Service génère un nouvel objet en utilisant l’interface « logState » pour le typage. La valeur « séquence number » de l’Event est assigné à l’attribut d’interface « séquence number », et la valeur « data » de l’Event qui contient le bloc de logs est assigné à l’attribut d’interface « log\_blocks » de typage tableau de string.  
Enfin, ce nouvel objet créé est stocké dans le tableau de la variable « complete log states ».

Une image contenant texte, Police, capture d’écran, ligne

Description générée automatiquementUne image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement   
S’il ne s’agit pas d’un nouveau test, le bloc de logs sont insérées dans le dernier objet du tableau de la variable « complete log states », à la suite du tableau contenant les blocs de logs précédents.  
  
Ainsi, chaque index du tableau de la variable « complete log states » contient un objet avec un « séquence number », ainsi qu’un tableau contenant un ou plusieurs transferts de bloc de lignes de logs.

Figure 13 Interface LogState

Figure 14 Méthode onEvent du Log Service

Afin de pouvoir fournir au composant les logs qui doivent être affichés, le service contient une variable « current log state » de typage « BehaviorSubjet ».   
  
En effet, lorsque l’utilisateur sélectionne un instant à afficher dans le composant de la barre de défilement du temps, un sequence number représentant la valeur sélectionnée est fournis au LogService, et la méthode « ManageCurrentState » récupére dans la variable « complete log states » l’objet LogState correspondant à ce séquence number.  
Cet objet contient les blocs de logs du test sélectionné, et est assigné à la variable « current log state », pour être affiché sur le composant.

### Architecture du composant

Une image contenant texte, Police, capture d’écran, ligne

Description générée automatiquementLe composant Angular « Log Component » est composé de trois fichiers. Un fichier TypeScript qui va contenir la logique du composant au sein de la classe LogsComponent, un fichier HTML pour définir la structure de l'interface utilisateur, et un fichier CSS pour gérer le style et la présentation de l'interface.  
  
Dans le constructeur de la classe LogsComponent, le logService est initialisé pour pouvoir établir une communication entre le composant et son service.  
La méthode OnInit est automatiquement exécuté lors de l’initialisation du composant. Cette méthode va s’abonner à la méthode getLogState du logService, qui va retourner en résultat la valeur de la variable « current log state » de ce service.  
Grace au fait que la variable « current log state » est un behaviorSubjet, il est possible d’être alerter à chaque changement de valeur grâce au principe d’abonnement. Ainsi, le fait d’assigner le résultat de cet abonnement à la variable « log\_blocks » permet d’avoir constamment l’information mis à jour en même temps que le service.

Figure 15 Fonctionnement de l'HTML du composant Logs

Figure 16 Fonctionnement du Typescript du Composant Log

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Il suffit ensuite d’afficher les blocs de logs dans la page Html en itérant le tableau de la variable « log\_blocks », et en plaçant le résultat dans une balise <pre> afin d’afficher les retours à la ligne « \n » entre chaque ligne de log.

## Problématique rencontrée et solution trouvée

### Identification de la problématique

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquementA la suite du développement de cette partie de l’application, je teste mon implémentation dans le but de chercher des erreurs ou des bugs éventuels que je n’ai pas encore corrigé.  
  
Je remarque que l’implémentation fonctionne et me livre le résultat attendu par le client lors de la définition du cahier des charges. Cependant, lorsque j’utilise l’application sur un fichier de log bien précis, et que j’affiche les lignes de log d’un test à un temps donné spécifique, le logiciel se met à se figer, puis à afficher un message d’erreur sans que je puisse en comprendre la raison. Malheureusement, ce message d'erreur n'est pas très parlant et n'aide pas à comprendre la raison exacte du dysfonctionnement.  
Après plusieurs recherches, je remarque que les lignes de logs contenues au sein de ce test sont du nombre de 500 000 lignes, ce qui est anormalement élevé comparé aux autres tests des fichiers mis à ma disposition.  
L’application arrive à gérer sans difficulté ni ralentissement cette quantité d’informations lors du parsing et du traitement d’informations sur le back-end, lors de l’envoi et à la réception Websocket sur le front-end, lors du traitement par le LogService, ainsi que lors du traitement sur la page Typescript du composant Log.  
L’entièreté de ce ralentissement vient du temps d’affichage au sein de la page Html, qui est beaucoup trop long. Le navigateur Chrome nous en informe avec ce message d’erreur.

Figure 17 Message d'erreur affiché sur le logiciel lors de l’affichage d’un test spécifique

Suite à la claire définition du besoin du client d'afficher en une seule fois l'ensemble des lignes de logs d'un test dans ce composant avec une barre de progression, tel que spécifié dans les points à réaliser du cahier des charges, et ne pouvant prendre la liberté de modifier le cahier des charges, j’ai convenu avec mon responsable de fixer un rendez-vous par visioconférence avec un responsable de l'équipe FUTE. L'objectif de cette réunion est que je puisse expliquer le problème que je rencontre et que nous puissions trouver des solutions de manière collaborative.

### Réunion client

Grace à la méthode Agile mise en place, et à l’interaction constante avec l’équipe FUTE, nous effectuons la réunion client avec le responsable de l’équipe le lendemain de la découverte de la problématique rencontré. Cette réactivité met permet de ne pas prendre de retard sur le planning attendu, tout en continuant à travailler dans de bonne condition sur des attentes clients bien définis.  
  
Lors de cette réunion, j’explique oralement la problématique rencontrée, en essayant d’adapter mon discours au maximum du au faite que le client ne connait pas le fonctionnement interne de l’application aussi bien que moi ou mon responsable.  
Après avoir expliquer la problématique ainsi que le point de blocage de l’application, nous nous mettons d’accord sur une redéfinition du résultat final attendu du composant log.

Désormais, le composant n’affichera plus l’intégralité des lignes de logs d’un test en un seul chargement, mais réalisera du « lazy loading », un chargement différé.  
L’affichage des lignes de log dans le composant sera défini par une quantité maximale, et l’utilisateur devra atteindre la dernière ligne de log affiché avec la barre de progression du composant pour charger les lignes de logs suivant.

### Nouvelle solution implémentée

Une fois cette réunion terminée et le fonctionnement final du composant redéfini avec le client, j’ai mis en application les décisions prise durant cet entretien en modifiant mon code existant.  
  
Afin de connaître le niveau de la barre de progression du composant, j’utilise la méthode « (scroll) » dans la balise <section> sur le fichier Html du composant. Ainsi, chaque fois que l'utilisateur fait défiler la section, cette méthode déclenche la fonction « onScroll » avec un évènement en argument contenant les informations de scroll.   
  
Dans cette fonction onScroll, je récupère trois valeurs :

« scrollHeight » qui contient la hauteur de la zone scrollable total,

« clientHeight » qui contient la hauteur de la zone affiché,

Et « scrollTop » qui contient la hauteur entre le haut de la zone scrollable total et le haut de la zone affiché, permettant de connaître la taille de toute la partie caché car déjà scrollé.

Si la taille de la partie cachée déjà scroller additionné à la hauteur de la zone affiché est égal à la taille de la zone total de la zone scrollable, alors nous somme à la fin de la zone scrollable. Il faut donc afficher le bloc de logs suivant en plus de celui déjà affiché.

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquementPour cela, une nouvelle variable, appelé « log\_blocks\_displayed » est incrémenté.

Figure 18 Méthode onScroll du Composant Log

Désormais, sur la page Html,

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

3.dire qu’on a des erreurs de code et qu’on les répare

2.Faire tests unitaires et tests manuelles(double)

3.Modifier du code

# Transformation du projet en livrable

## Création d’un Dockerfile

4.Ecrire un Dockerfile

## Création d’une documentation technique

4.faire une doc qui n’exister pas encore sur un logiciel/composant

## Validation du livrable final par le client

2.Se faire valider ses livrable par le client

# Projet personnel : Outil de sauvegarde et de suivis des analyses d’eau

## Présentation du projet

1.Aveugle peut aller sur le site(double)

## Cahier des charges

1.Faire de la cybersécurité, rendre son app sécurisé (.env, mdp hashé)

1.Faire du code propre(triple)

1.Cacher l’accès à la bdd avec un fichier .env, Verif input dans Controller et push bdd(double)

## Conception UML

1.Diagramme usecase/activité/sequence

1.Cacher l’accès à la bdd avec un fichier .env, Verif input dans Controller et push bdd(double)

## Conception MCD et MLD

1.Utiliser une bdd qui peut se connecter a n’importe quel app

2.faire un document qui explique les nomenclatures d’une bdd

4.Cacher des données en bdd, hasher password

## Maquettage

1.Faire des maquettes sur figma

## Architecture logicielle

1.Concevoir service et composant réutilisable(double)

2. avoir architecture mvc ou service composant pour être au max réutilisable(double)

3.passer d’un document diagramme à du code

## Tests Unitaires

2.Faire tests unitaires et tests manuelles(double)

3. faire des tests unitaires

## Déploiement sur serveur

1.Prendre en compte que l’app soit avoir plusieurs utilisateurs en simultanés

# Conclusion

# Bibliographie et citations

|  |  |
| --- | --- |
| [1] | V. GARDEAU, «CELAD,» Toulouse. |

# Glossaire

# Table des illustrations

[Figure 1 Chiffre d’affaires et collaborateurs 7](file:///C:\xampp\htdocs\licence3\mémoire\brouillon2_memoire.docx#_Toc141317832)

[Figure 2 Interaction entre le TECU et le logiciel FUTE 8](file:///C:\xampp\htdocs\licence3\mémoire\brouillon2_memoire.docx#_Toc141317833)

[Figure 3 Exemple de ligne de log GUI de type signal update 10](file:///C:\xampp\htdocs\licence3\mémoire\brouillon2_memoire.docx#_Toc141317834)

[Figure 4 Parcours utilisateur de l'application 11](file:///C:\xampp\htdocs\licence3\mémoire\brouillon2_memoire.docx#_Toc141317835)

[Figure 5 Composant d'affichage des logs 11](file:///C:\xampp\htdocs\licence3\mémoire\brouillon2_memoire.docx#_Toc141317836)

[Figure 6 Barre de défilement du temps 11](file:///C:\xampp\htdocs\licence3\mémoire\brouillon2_memoire.docx#_Toc141317837)

[Figure 7 Architecture logicielle Back-end 12](file:///C:\xampp\htdocs\licence3\mémoire\brouillon2_memoire.docx#_Toc141317838)

[Figure 8 Architecture Front-end du stockage des Events 13](file:///C:\xampp\htdocs\licence3\mémoire\brouillon2_memoire.docx#_Toc141317839)

[Figure 9 Architecture Front-end de l'affichage d'un Event 13](file:///C:\xampp\htdocs\licence3\mémoire\brouillon2_memoire.docx#_Toc141317840)

[Figure 10 Planning de la réalisation du développement de l'application 14](file:///C:\xampp\htdocs\licence3\mémoire\brouillon2_memoire.docx#_Toc141317841)

[Figure 11 Composant d'affichage des logs 16](file:///C:\xampp\htdocs\licence3\mémoire\brouillon2_memoire.docx#_Toc141317842)

[Figure 12 Ligne de log GUI signifiant le déclenchement d'un nouveau test 17](file:///C:\xampp\htdocs\licence3\mémoire\brouillon2_memoire.docx#_Toc141317843)

[Figure 13 Interface LogState 18](file:///C:\xampp\htdocs\licence3\mémoire\brouillon2_memoire.docx#_Toc141317844)

[Figure 14 Méthode onEvent du Log Service 18](file:///C:\xampp\htdocs\licence3\mémoire\brouillon2_memoire.docx#_Toc141317845)

[Figure 16 Fonctionnement de l'HTML du composant Logs 19](file:///C:\xampp\htdocs\licence3\mémoire\brouillon2_memoire.docx#_Toc141317846)

[Figure 15 Fonctionnement du Typescript du Composant Logs 19](file:///C:\xampp\htdocs\licence3\mémoire\brouillon2_memoire.docx#_Toc141317847)

# Annexes