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

[II. Tableau des compétences 4](#_Toc143370291)

[III. Présentation Personnelle 6](#_Toc143370292)

[IV. Présentation de l’entreprise 6](#_Toc143370293)

[A. Historique 6](#_Toc143370294)

[B. Activités et Chiffres-clé 7](#_Toc143370295)

[C. Mon équipe et mes activités 7](#_Toc143370296)

[V. Présentation du client 8](#_Toc143370297)

[A. Équipe et activités 8](#_Toc143370298)

[B. Besoins exprimés 9](#_Toc143370299)

[VI. Définition du projet 10](#_Toc143370300)

[A. Cahier des charges 10](#_Toc143370301)

[B. Architecture des interfaces 11](#_Toc143370302)

[Architecture logicielle 12](#_Toc143370303)

[1. Back-end 12](#_Toc143370304)

[2. Front-end 13](#_Toc143370305)

[VII. Méthodologie du projet 14](#_Toc143370306)

[A. Planning 14](#_Toc143370307)

[B. Logiciels de travail collaboratif 14](#_Toc143370308)

[C. Méthode Agile 15](#_Toc143370309)

[D. Réunions client 15](#_Toc143370310)

[VIII. Implémentation du composant log et de son service 16](#_Toc143370311)

[A. Contexte et objectif 16](#_Toc143370312)

[B. Interface du composant 16](#_Toc143370313)

[C. Mise en œuvre et réalisation 17](#_Toc143370314)

[1. Récupération de la donnée 17](#_Toc143370315)

[2. Architecture du service 18](#_Toc143370316)

[3. Architecture du composant 19](#_Toc143370317)

[D. Problématique rencontrée et solution trouvée 20](#_Toc143370318)

[1. Identification de la problématique 20](#_Toc143370319)

[2. Réunion client 21](#_Toc143370320)

[3. Nouvelle solution implémentée 21](#_Toc143370321)

[IX. Transformation du projet en livrable 23](#_Toc143370322)

[A. Création d’un Dockerfile 23](#_Toc143370323)

[B. Validation du livrable par le client 29](#_Toc143370324)

[C. Maintenance du projet 29](#_Toc143370325)

[X. Projet personnel : Outil de sauvegarde et de suivis des analyses d’eau 30](#_Toc143370326)

[A. Présentation du projet 30](#_Toc143370327)

[B. Cahier des charges 31](#_Toc143370328)

[C. Plan de navigation utilisateur 31](#_Toc143370329)

[D. Conception UML 32](#_Toc143370330)

[1. Diagramme de cas d’utilisation 32](#_Toc143370331)

[2. Diagramme d’activité 33](#_Toc143370332)

[3. Diagramme de séquence 34](#_Toc143370333)

[4. Diagramme de classe 35](#_Toc143370334)

[E. Conception MCD et MLD 36](#_Toc143370335)

[1. Modèle conceptuel de données 36](#_Toc143370336)

[2. Modèle logique de données 37](#_Toc143370337)

[F. Maquettage 38](#_Toc143370338)

[Wireframes 38](#_Toc143370339)

[1. Thème 39](#_Toc143370340)

[2. Mockups 40](#_Toc143370341)

[G. Architecture 41](#_Toc143370342)

[1. Architecture logicielle 41](#_Toc143370343)

[2. Arborescence des fichiers 42](#_Toc143370344)

[H. Tests Unitaires 43](#_Toc143370345)

[I. Cybersécurité 44](#_Toc143370346)

[1. Sécurisation de la base de données 44](#_Toc143370347)

[2. Hachage des mots de passe utilisateurs 46](#_Toc143370348)

[J. Déploiement 48](#_Toc143370349)

[1. Déploiement Serveur 48](#_Toc143370350)

[2. Déploiement de la base de données 49](#_Toc143370351)

[XI. Conclusion 50](#_Toc143370352)

[1. Conclusion professionnelle 50](#_Toc143370353)

[2. Conclusion personnelle 51](#_Toc143370354)

[3. Perspective future 52](#_Toc143370355)

[XII. Bibliographie et citations 53](#_Toc143370356)

[XIII. Glossaire 53](#_Toc143370357)

[XIV. Table des illustrations 53](#_Toc143370358)

[XV. Annexes 54](#_Toc143370359)

# 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. | Conception UML |
| 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  Architecture:Architecture logicielle  Architecture:Arborescence des fichiers |
| Concevoir des services d’accès aux données indépendants du mode de stockage en garantissant la sécurité des données pour produire du logiciel partageable. | Architecture:Architecture logicielle  Architecture:Arborescence des fichiers |
| 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. | Déploiement:Déploiement Serveur |
| 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. | Maquettage:Wireframes  Maquettage:Mockups |
| Identifier les risques et leur niveau de criticité pour permettre leur prévention. | Cybersécurité:Sécurisation de la base de données  Cybersécurité:Hachage des mots de passe utilisateurs |
| 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  Projet personnel : Outil de sauvegarde et de suivis des analyses d’eau:Cahier des charges |
| 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. | Cybersécurité:Sécurisation de la base de données  Cybersécurité:Hachage des mots de passe utilisateurs |
| 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. | Conception UML |
| 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. | Conception MCD et MLD |
| 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  Architecture:Architecture logicielle  Architecture:Arborescence des fichiers |
| 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. | Transformation du projet en livrable:Validation du livrable par le client |
| 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. | Tests Unitaires |
| 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. | Cybersécurité:Hachage des mots de passe utilisateurs  Déploiement:Déploiement de la base de données |
| Modifier un algorithme sans générer de dysfonctionnements en comprenant et s’adaptant si besoin au mode de pensée de son auteur. | Problématique rencontrée et solution trouvée:Nouvelle solution implémentée |
| 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. | Problématique rencontrée et solution trouvée:Nouvelle solution implémentée |
| 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. | Tests Unitaires |
| 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. | Conception MCD et MLD:Modèle conceptuel de données  Cybersécurité:Hachage des mots de passe utilisateurs |
| 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. | Transformation du projet en livrable:Création d’un Dockerfile |

# 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 cette 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

Au milieu de mon d’année d’alternance, 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.

En 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.

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

Description générée automatiquementTECU 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 dites « 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.

Figure 7 Architecture logicielle Back-end

La valeur du « séquence number » est ensuite généré en réalisant 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".

### 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 automatiquementL’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 ».

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

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.

Cette architecture de stockage des données permet de ne pas devoir récupérer les données dans le fichier log à chaque sélection d’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

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.

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

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.  
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’avancée 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 planifié après une livraison de l’application sous format de container 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 définit 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.

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

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.

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 automatiquementS’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.

Figure 15 Fonctionnement de l'HTML du composant Logs

Figure 16 Fonctionnement du Typescript du Composant Log

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.

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

Description générée automatiquementIl 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.

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

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.

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 expliqué 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,

« 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 additionner 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, en se servant de cette nouvelle variable ainsi que de l’index des blocs de logs à afficher, il m’est possible de faire apparaître les lignes de logs en lazy loading.

Ainsi, le temps de chargement de l’affichage des lignes de logs est limité au premier bloc de lignes de log, et le fait de scroller jusqu’en bas du composant permet à l’utilisateur d’afficher un nouveau bloc additionnel.

Figure 19 Nouvelle implémentation de l'HTML du composant Logs

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

Description générée automatiquementUne fois cette implémentation réalisée, je réalise différents tests manuels avec plusieurs fichiers de logs différents, dans l’objectif de vérifier que la fonctionnalité fonctionne correctement, et pour pouvoir détecter en amont tous les bugs éventuels.

Une fois que tout est opérationnel, je pousse mes modifications sur le Gitlab de Renault afin de sauvegarder mon travail.

Je réalise ensuite une demande de merge-request de ma branche, sur lequel mon responsable, ainsi que le responsable de l’équipe FUTE pourront relire mon implémentation et faire une demande de modification si nécessaire.

Dans mon cas, il n’y a pas eu de demande de vérification de leur part, et la marge request a été accepté.

# Transformation du projet en livrable

La livraison de cette application est réalisée sous format de container docker, afin d’être facilement transportable et déployable par l’équipe FUTE. Elle est effectuée en deux temps.

La première livraison, réalisé à la fin de la première moitié de l’implémentation du projet, a comme objectif de récupérer des expériences utilisateurs le plus rapidement possible afin d’adapter au mieux le projet à leurs besoins.

La seconde livraison représente le déploiement final de l’application. C’est celle qui sera déployé chez le client.

## Création d’un Dockerfile

Pour pouvoir réaliser le container docker, je conçois un dockerfile. Il s’agit d’un fichier de script basé sur une image déjà existante qui sert à transporter et déployer facilement la configuration d’un environnement, ainsi que son éventuel contenu sous forme de chemin, comme une application par exemple.

En exécutant un dockerfile, on construit une image docker. Il s’agit d’un modèle exécutable statique contenant tout le code, les bibliothèques ainsi que les dépendances.

Cette image sert par la suite à déployer un container docker, semblable a une machine virtuelle légère. C’est ce container qui déploie l’application pour la rendre disponible pour les utilisateurs.

L’application Web que je développe pour l’équipe FUTE contient une première partie d’application frontend en Angular, ainsi qu’une seconde partie d’application backend en Python. Ces deux parties d’application ont des besoins de configurations d’environnements très différentes.

Afin de répondre à ces besoins spécifiques pour faire fonctionner mon application, je dois donc concevoir deux dockerfiles pour chacune de ces parties.

Cette approche me permet de maintenir des environnements séparés, ce qui facilite grandement la gestion et la maintenance de l'application.

Lorsque plusieurs dockerfiles doivent être déployés ensemble, comme c’est mon cas ici, la gestion de ce déploiement s’effectue grâce à un nouveau fichier appelé un docker-compose. C’est un fichier qui permet de gérer les applications multi-containers, et de spécifier les services, réseaux et volumes spécifique pour l’application. Il facilite la liaison ainsi que la communication entre ces différents containers.  
  
Voilà la réalisation de mes dockerfiles et docker-compose étape par étape.

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

Description générée automatiquementPour commencer, je réalise le dockerfile destiné à la partie backend de l’application.

Figure 20 Dockerfile Backend

Le backend est développé en langage python de version 3.8.0 J’utilise donc une image python3.8.0 pour concevoir ce dockerfile.

Je crée un nouveau dossier appelé « app » à la racine du container. Je copie ensuite dans le dossier « app » l’intégralité des fichiers contenus dans le répertoire courant. Il s’agit de l’intégralité de l’application backend.

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

Description générée automatiquementJe réalise l’installation de toutes les dépendances nécessaires en exécutant la ligne de commande « pip install -r requirement.txt ». Le fichier requirement.txt représente une liste de l’ensemble des dépendances pour faire fonctionner l’application backend. C’est un fichier que j’ai mis à jour durant toute la durée du développement.

Figure 21 Fichier requirement.txt

J’indique ensuite que j’expose le port 5001. Il ne s’agit pas d’une commande pour exposer le port mais plus d’une documentation visant à indiquer aux futur développeur le port sur lequel l’application est exposée.

J’indique ensuite un ordre de commande « python app.py », qui permet de lancer l’application. Cette commande ne sera pas réalisé lors de la création de l’image mais à l’activation du container.  
Une fois le dockerfile terminé, il me suffit ensuite de réaliser la commande « docker build -t imagefdmtbackend . » à la racine de ce dockerfile pour créer l’image docker en lien avec ce dockerfile.

La commande « docker run -d -p 5001:5001 --name fdmtbackend imagefdmtbackend » me permet ensuite d’utiliser l’image nouvellement créé pour construire mon docker, et le faire fonctionner.

Une fois cela réalisé, je peux constater que mon application backend est disponible et fonctionnelle en réalisant des requêtes APIs vers mon container.

Je passe ensuite à la conception du dockerfile destiné à la partie frontend de l’application.

Une image contenant texte, capture d’écran, Police

Description générée automatiquementPour la réalisation du dockerfile de cette partie frontend en Angular, j’ai besoin de deux environnements. Le premier est un environnement de construction, servant à builder l’application, et le second est un environnement servant à déployé l’application.  
Pour l’environnement de construction, je commence par utiliser une image de node :18, car il s’agit de la version de node la plus haute qui reste fonctionnelle avec ma version 15 de Angular. [2]

Figure 22 Dockerfile frontend

Je nomme ce nouvel environnement « build », puis je crée un nouveau dossier appelé « app » à la racine du container. Je copie ensuite dans ce dossier l’ensemble des fichiers et dossiers contenus dans le même répertoire que le dockerfile. Il s’agit de l’intégralité de l’application frontend Angular. Cependant, je ne souhaite pas copier certains dossiers, comme node module par exemple.

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

Description générée automatiquementPour éviter de copier certains fichiers ou dossiers, j’utilise un fichier appelé « .dockerignore ». Il me permet d’ignorer l’ensemble des fichiers et dossiers non nécessaires au déploiement de l’application. Ce fichier regroupe les mêmes fichiers et dossiers que le fichier .gitignore classique d’un projet Angular.

Figure 23 Dockerignore du dockerfile frontend

Une fois les fichiers et dossiers copiés, je réalise la commande « npm install ». Cette commande va télécharger l’ensemble des dépendances du fichier package.json ainsi que package-lock.json, afin de créer le dossier node module contenant toutes ces dépendances.

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

Description générée automatiquementJe réalise ensuite la commande « npm run build ». Cette commande va créer une version de production de l’application. Il s’agit d’une version compilée de l’application qui sert à minimiser et optimiser l’application une fois déployé. Cette version compilée est contenue dans le dossier « app/dist/fdmtfrontend ».

Voici un exemple de cette version compilé que j’ai réalisé en local.

Figure 24 Version de production du projet angular

Mon application est désormais en version compilé, je passe donc à l’environnement de déploiement. Pour que la partie frontend du projet soit disponible en requête HTTP, elle nécessite d'être hébergée sur un serveur Web tel que Nginx. Il s’agit d’un serveur léger servant à héberger des sites Web, des applications Web et des services en ligne.

J’utilise donc une image de nginx pour le déploiement.

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

Description générée automatiquementJe supprime la configuration par défault de nginx en réalisant la commande « rm /etc/nginx/conf.d/default.conf », puis je copie le fichier « nginx.conf » sur le chemin « «/etc/nginx/conf.d » , en utilisant la commande « COPY nginx.conf /etc/nginx/conf.d ».

Figure 25 Fichier nginx.conf du dockerfile frontend

Le fichier nginx.conf est un fichier de configuration du serveur nginx que j’ai configuré pour être adapter à mon projet.

J’indique que sur la racine du serveur, les fichiers sont situés dans le chemin « /user/share/nginx/html », et que à la connexion, le serveur doit chercher le ficher « index.html »

Dans le fichier Dockerfile, je spécifie que je souhaite copier les fichiers de l'emplacement « /app/dist/fdmtfrontend » de l'environnement de construction où se trouve la version compilée de l'application Angular, pour pouvoir les copier vers le répertoire "/usr/share/nginx/html" de l'environnement Nginx, où le serveur Web va les déployer.

J’indique ensuite que j’ouvre le port 80, ou mon application sera accessible.

Le dockerfile de la partie frontend terminé, je peux désormais construire mon image avec la commande « docker build -t imagefdmtfrontend . » exécuté dans le répertoire actuel ou se trouve mon dockerfile. Je peux ensuite construire mon container avec la commande « docker run -d -p 8080:80 --name fdmtfrontend imagefdmtfrontend », ou j’indique que l’application est disponible sur le port 8080.

Il me suffit ensuite d’aller vérifier sur un navigateur internet l’url « http://localhost:8080 » pour vérifier que mon container fonctionne correctement.

Mon dockerfile de la partie frontend ainsi que mon dockerfile de la partie backend de mon application sont désormais fonctionnels et testé.

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

Description générée automatiquementJe peux donc passer à la création de mon fichier docker-compose qui va gérer ces deux dockerfiles, les interactions entre eux ainsi que les volumes nécessaires. Je décide de créer ce docker-compose dans le dossier qui contient les deux parties de l’applications avec leur dockerfile respectif.

Figure 26 Docker compose

Sur ce docker compose, je commence par indiquer la version de syntaxe 3.8 du docker compose.

Je définie ensuite mes services. Chaque service va créer un container.

Le premier service est appelé fdmtbackend et va déployer le container contenant la partie backend de l’application. J’indique que ce container doit s’appeler fdmtbackend avec la commande « container\_name :fdmtbackend ».

J’indique que le dockerfile qui doit être utiliser pour la création de ce container est situé dans le dossier fdmtbackend avec la comande « build: ./fdmtbackend ».

Je connecte ensuite le port 5001 du serveur ou sera exécuté ce docker compose au port 5001 du container.

Enfin, je crée un volume. Il s’agit d’une copie d’un dossier en temps réel entre un emplacement sur le serveur ou est déployer le docker compose et un emplacement à l’intérieur du container. Cela permet de faire transiter des fichiers du serveur au container et inversement. La commande « - /tmp/FUTE\_Logs:/tmp/FUTE\_Logs » indique que je souhaite créer le volume entre le chemin /tmp/FUTE\_Logs du serveur et le chemin /tmp/FUTE\_Logs du container.

Je passe désormais au second service appelé fdmtfrontend. Il va déployer le container contenant la partie frontend de l’application. J’indique que ce container doit s’appeler fdmtfrontend avec la commande « container\_name :fdmtfrontend ».

J’indique que le dockerfile qui doit être utiliser pour la création de ce container est situé dans le dossier fdmtfrontend avec la comande « build: ./fdmtfrontend ».

Je connecte ensuite le port 8080 du serveur ou sera exécuté ce docker compose au port 80 du container.

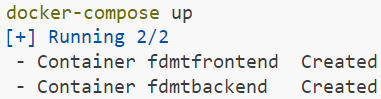
Les deux services maintenant créés dans le docker compose, j’exécute ce fichier avec la commande « docker-compose up » afin de vérifier que tout fonctionne correctement.

Figure 27 Résultat du docker-compose

L’application est disponible sur mon navigateur à l’url « http://localhost:8080 », et le frontend arrive à se connecter avec le backend en passant par le port 5001. Le backend a accès au volume précédemment créé. Le docker-compose est donc fonctionnel.

## Validation du livrable par le client

Durant la conception du projet, j’ai effectué deux types de validation de livrable par le client.

La première validation de livrable est une validation continue. En effet, à chaque nouvelle implémentation, une validation est réalisée lorsque le responsable de l’équipe FUTE approuve ma demande de merge requête. Cette validation permet un suivi continue du projet d’un point de vue technique pour l’équipe FUTE, et permet de détecter en amont les problèmes éventuels.

La seconde validation de livrable a été effectué sous format de container docker, en deux étapes différentes.

La première livraison en format de container docker a été réalisé au milieu du développement de ce projet. L’objectif est que je récupère des retours d’expériences utilisateurs le plus tôt possible, afin que je puisse les prendre en compte lors de la suite de l’implémentation du projet.

La seconde livraison sous format de container a été réalisé à la fin du projet. Juste avant cette livraison, j’ai réalisé une présentation globale de l’application ainsi que de l’ensemble des interfaces au responsable de l’équipe FUTE, afin qu’il valide le résultat d’implémentation de chaque interface.

J’ai ensuite effectué la seconde livraison, avec comme objectif de faire tester l’application aux utilisateurs afin de rechercher des éventuels bugs et anomalies qui serait encore sur l’application avant la validation finale du client.

Dans mon cas, il n’y a pas eu de bugs détectés. La livraison finale est donc considérée terminée, et la validation finale du client est donnée, faisant preuve de déclenchement pour la dernière facturation du projet.

## Maintenance du projet

Sur ce projet, une facturation sous forfait a été réalisé. Cela signifie que la facturation regroupe l’ensemble du travail effectuer de la conception jusqu’à la livraison finale.

Cependant, une fois cette livraison terminé et validé par les responsable FUTE, toute modification ou ajout de nouvelle fonctionnalité supplémentaire qui demande à faire appel à l’équipe CAT doit se faire systématiquement sur une nouvelle facturation.

Ainsi, sans facturation, l’ensemble de la maintenance de ce projet doit être réalisé par l’équipe FUTE. L’équipe CAT reste néanmoins disponible pour toute nouvelle implémentation avec nouvelle facturation si nécessaire, et peut ainsi faire profiter de l’expérience qu’elle a prise sur le projet pour faciliter le développement de nouvelle fonctionnalité.

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

## Présentation du projet

Pendant ma deuxième année de licence, j'ai eu pour projet de choisir un besoin utilisateur, concevoir une application Web pour y répondre, puis la développer.

J'ai décidé de combler un besoin personnel qui concerne le stockage et le suivi des données relatives aux minéraux contenus dans l'eau de mon aquarium.

Afin de répondre à ce besoin, il faut que l’application que je vais concevoir ait un système de création de compte et de connexion utilisateur, ce qui permet d’avoir un espace de travail individuel.

Il faut que chaque utilisateur puisse avoir sa propre une zone de stockage d’informations pour chacun de ces aquariums si celui-ci en possède plusieurs, et la possibilité de rajouter ou supprimés des types de minéraux pour chaque aquarium sur lequel il souhaite réaliser des analyses.

Enfin, il faut qu’il puisse insérer des données à une date sélectionné, et que l’information soit stockée par l’application, pour pouvoir être ensuite affiché et analysé par l’utilisateur sous format d’un tableau ou de graphique.

L’objectif final étant de donner à l’utilisation la possibilité d’avoir une vision global des informations sur le long terme, et de pouvoir faire parler les données pour en tirer des conclusions.

## Cahier des charges

Pour concevoir cette application, je réalise un cahier des charges afin de mieux clarifier les objectifs, structurer le projet et éviter les oublis qui pourrait causer des problèmes lors du développement.

Les technologies que j’ai sélectionné pour cette réalisation sont l’Html, le Css et le Javascript pour la partie Front-end de l’application.

Pour la partie Back-end, j’ai sélectionné le langage PHP, dans le but de facilité le déploiement et de diminuer les frais de serveur qui seront financés personnellement.

La base de données sera relationnelle et sera créer en MySql.

L’application doit être conçu de manière à avoir un code propre, facilement maintenant, et évolutif. Une attention particulière sera apportée à la cybersécurité, que ce soit sur la sécurité des mots de passe utilisateur qui ne doivent pas être visible en base de données, ou bien sur l’environnement de travail en général.

Le design du projet doit être adaptatif pour permettre une expérience utilisateur fluide sur différentes plateformes, notamment les ordinateurs de bureau, les tablettes et les smartphones.

Enfin, le logiciel doit être conçu de manière à être facilement déplaçable de l’environnement de développement à l’environnement de déploiement, dans un principe d’intégration continue.

## Plan de navigation utilisateur

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

Description générée automatiquementVoici le plan de navigation utilisateur de l’application.

Figure 28 Plan de navigation utilisateur

Une fois arrivé sur l’application, l’utilisateur arrive sur la page « homepage », qui lui permet de choisir entre la création de son compte, ou la connexion si celui-ci a déjà un compte. S’il souhaite créer un compte, la page « register » lui permet de s’enregistrer. S’il souhaite se connecter, la page « login » lui permet de s’identifier.

Une fois connecté, il accède à la page « values\_insertion » pour insérer des données, la page « data\_charts » pour afficher les données sous un format de graphiques, et la page « data\_table » pour afficher les données sous un format de tableau.

## Conception UML

Avant de débuter le développement, j’ai réalisé des diagrammes UML afin de bien me représenter les fonctionnalités de l’application et aider à leurs développements.

### Diagramme de cas d’utilisation

Le diagramme de cas d’utilisation permet de définir tous les cas d’utilisation de l’application, les acteurs, ainsi que les droits de chacun.

Pour cette application, le diagramme contient deux acteurs :

Le premier représente le Visiteur. Il possède le cas d’utilisation d’inscription ainsi que le cas d’utilisation de connexion.

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

Description générée automatiquementLe second acteur représente l’Utilisateur. Il hérite de tous les cas d’utilisation du Visiteur car il représente l’acteur Visiteur une fois la connexion effectuée. Il possède aussi ces propres cas d’utilisation, qui ne sont pas disponible pour le Visiteur.

Figure 29 Diagramme des cas d'utilisation

Dans le but d'améliorer la compréhension des différents cas d'utilisation entre les pages, j’ai attribué une couleur spécifique à chaque page de l'application, ainsi qu’à la barre de navigation. Cette approche permet de mieux définir les cas d'utilisation associés à chacune des pages.

### Diagramme d’activité

Le diagramme d'activité est un processus algorithmique qui permet de réfléchir de manière algorithmiquement sur comment se comporte la fonctionnalité.

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

Description générée automatiquementLe diagramme présenté ci-dessous représente le cas d’utilisation de création de compte. Il contient trois « swimlanes », représentant les trois intervenants lors du processus de création du compte, que sont le Visiteur, le système d’information ainsi que la base de données.

Figure 30 Diagramme d'activité de création de compte

Comme nous pouvons le voir dans ce diagramme, le « Accept signal » est déclenchée par le Visiteur lorsqu’il clique sur le bouton s’inscrire. Le système d’information affiche alors la page « register », afin que le visiteur puisse insérer les informations d’inscription. Une fois validé, le système d’information vérifie les différentes insertions réalisées par le visiteur, puis vérifie si l’email n’existe pas déjà en base de données. Il hashe ensuite le mot de passe, puis crée l’enregistrement sur la table user dans la base de données. Enfin, il redirige vers la page « Homepage ».

Les différentes erreurs possibles sont capturées puis renvoyées sur la page « register » où le visiteur pourra en prendre connaissance. Si l’erreur est considérée non bloquante, le visiteur pourra de nouveau réinsérer les informations d’inscription. Sinon, il s’agit d’un « Flow final ».

### Diagramme de séquence

Le diagramme de séquence est un processus qui permet de réfléchir de manière chronologique sur comment se comporte la fonctionnalité.

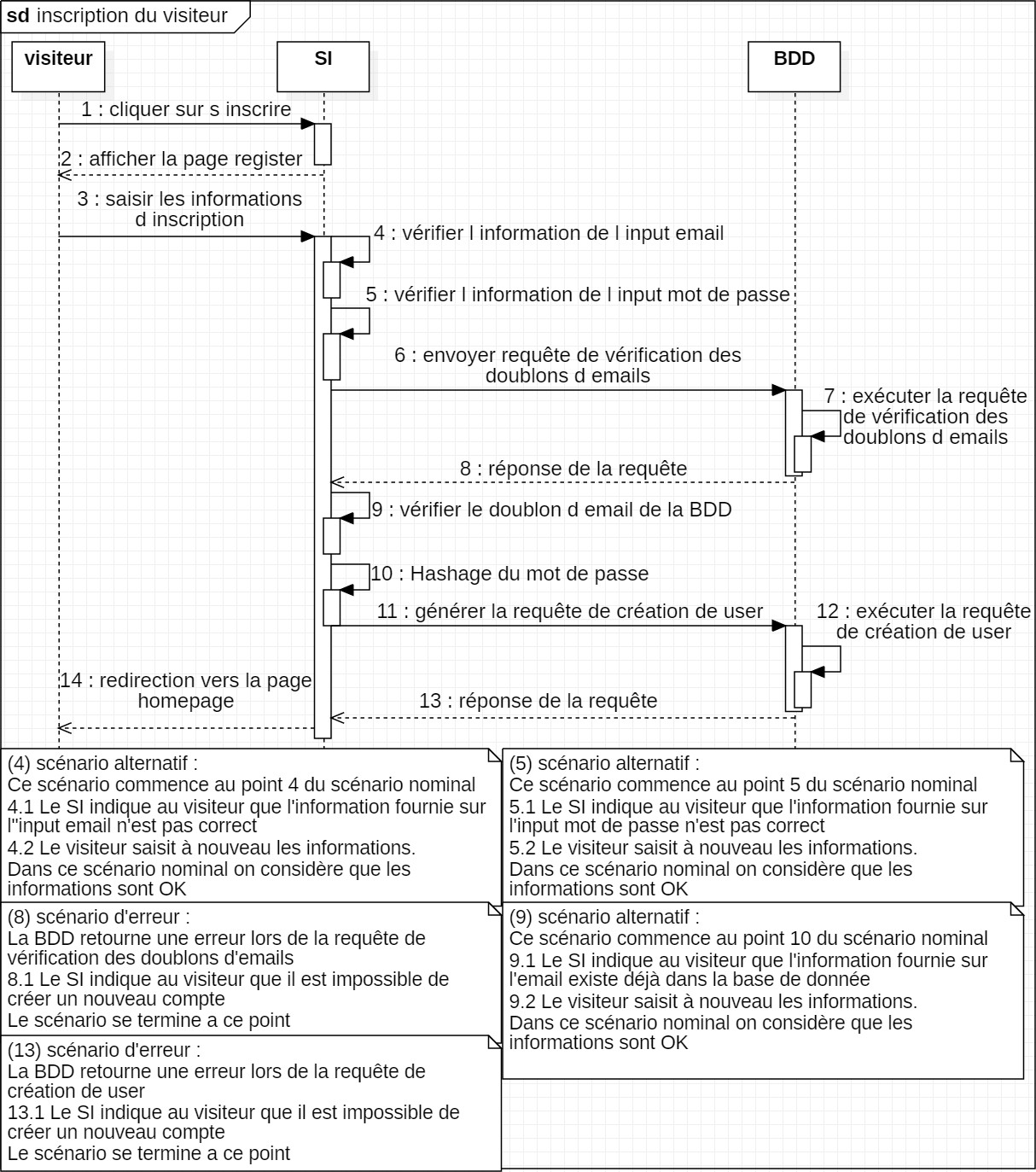
Le diagramme présenté ci-dessous représente lui aussi le cas d’utilisation de création de compte. Il contient les trois mêmes intervenants que sur le diagramme d’activité, avec pour chacun un « Lifeline », ou ligne de vie.

Figure 31 Diagramme de séquence de création de compte

Comme nous pouvons le voir, seul le scénario initial est schématisé dans ce diagramme de séquence.  
Les scénarios alternatifs qui représente les scénarios non bloquants, ainsi que les scénarios d’erreur, qui représente les scénarios bloquants, sont affichés sous forme de note séparé en dessous du diagramme de séquence.

Ces notes séparées expliquent en détails les différents scénarios sous forme d’étapes jusqu’au retour au scénario initial, ou jusqu’à la fin d’un scénario.

### Diagramme de classe

Le diagramme de classe représente la façon dont les données vont être gérés et comment nous pourrons accéder à ces données.

Une image contenant texte, reçu, Police, nombre

Description générée automatiquementNous pouvons voir sur ce diagramme les trois classes « User », « Aquarium » et « Types\_analysis », qui représente pour chacun leur table respective en base de données.

Figure 32 Diagramme de classes des entités

Chacune de ces classes contiennent des attributs privés, ainsi que leurs accesseurs et mutateurs permettant d’accéder à ces attributs.

La nomenclature du nom de ces classes est en PascalCase, tandis que les attributs, les accesseurs et les mutateurs sont en snake\_case.

Il est aussi présenté sur ce diagramme les différentes relations de classes entre « User » et « Aquarium », ainsi qu’entre « Aquarium » et « Types analysis ».

Les instances générées par ces classes sont gérées par des services, appelés dans cette application des « repository ».

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

Description générée automatiquementNous pouvons voir sur ce second diagramme les trois services « UserRepository », « AquariumRepository » et « TypeAnalysisRepository », qui représente pour chacun le service dédié aux instances de chacun des tables de la base de données.

Figure 33 Diagramme de classes des services

Sur ces services, l’attribut privé « database » et son mutateur « set\_database » sont en snake\_case, tandis que les méthodes qui gère les instances de classes sont en CamelCase.

## Conception MCD et MLD

### Modèle conceptuel de données

Le Modèle Conceptuel de Données (MCD) est une modélisation de la base de données de l’application, ce qui permet d’avoir une vue complète de l’ensemble de la base.

C’est lors de la réalisation du MCD que je conceptualise les entités, leurs attributs ainsi que leurs cardinalités, en liens avec les diagrammes de cas d’utilisation, d’activités et de séquences préalablement réalisés.

Une image contenant texte, capture d’écran, Post-it, diagramme

Description générée automatiquementVoici le modèle conceptuel de données pour cette application :

Figure 34 Modèle Conceptuel De Données

Nous pouvons voir la table « users » qui sert à contenir les différents enregistrements de l’acteur « Visiteur » lorsqu’il réalise une création de compte, comme indiqué dans le diagramme de cas d’utilisation.

C’est dans cette table que les données qui permette la connexion sont stockées, telle que l’email qui est enregistré dans l’attribut « email\_user », ou le mot de passe qui est au préalablement hashé avant d’être enregistré dans l’attribut « password\_user ». Il y a aussi un attribut « id\_user » qui contient un id unique créé à chaque nouvel enregistrement.

Nous pouvons aussi voir les différentes cardinalités entre les tables, afin de mieux comprendre les relations entre elles.

Par exemple, la cardinalité entre la table « users » et « aquariums », qui est représenté par le verbe à l’infinitif « posséder », est une relation one to many. Un enregistrement de la table « users » peut donc être en lien avec zéro ou plusieurs enregistrements de la table « aquariums », mais un enregistrement de la table « aquariums » doit être en lien qu’avec un seul enregistrement de la table « users ».  
  
Afin de faciliter le développement et la compréhension, je défini une nomenclature strict pour le nommage des table, qui sont en Snake Case et à l’infinitif, ainsi que pour les attributs, qui sont aussi en Snake Case, avec le nom de la valeur suivis du nom de la table.

### Modèle logique de données

Le Modèle Logique de Données (MCD) est une modélisation de la base de données orienté pour faciliter le développement. Son objectif est de modélisé la base de la manière la plus semblable à son résultat final lors de la création des tables et des relations.

En comparaison au MCD, les verbes à l’infinitif sur les cardinalités disparaissent.  
Les types de relations, affichés précédemment par des 0, 1 et N sont modifiés pour être remplacés par des clés étrangères et des tables d’associations.

Par exemple, la relation one to many entre « users » et « aquariums » se transforme en une nouvelle colonne « id\_user » au sein de la table « aquariums ». Cette nouvelle colonne sert de clé étrangère en lien avec la clé primaire « id\_user » de la table « users » pour faire le lien entre les deux tables.

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

Description générée automatiquementCette application ne nécessite pas de créer de relations many to many. Il n’y a donc pas de table d’association dans cette base.

Figure 35 Modèle Logique de Données

Durant la réalisation, je choisi volontairement de créé des relations qui crée des suppressions en cascade, avec les relations en 1,1 du MCD.

En effet, si l’utilisateur supprime un aquarium, je souhaite que les types d’analyses, les valeurs de types d’analyses ainsi que les commentaires qui sont en relation avec cet aquarium soit aussi supprimé, car ils n’ont plus d’utilités dans l’application.

De même, si un utilisateur est supprimé, je souhaite que tous les aquariums en relation avec cet utilisateur soient eux aussi supprimés.

## Maquettage

Pour la réalisation du maquettage de l’application, je privilégie le format mobile, afin de rendre le logiciel le plus responsive possible.

Wireframes  
Je commence par réaliser un Wireframe pour chaque interface directement sur une feuille de papier, afin d’avoir rapidement une vue d’ensemble des différents cas d’utilisation, leur disposition sur l’application, ainsi que leurs interactions les uns avec les autres.

L’objectif de cette démarche est que je travail de manière rapide et efficace, et que je réalise plusieurs brouillons afin d’optimiser la navigation et l’expérience utilisateur (UX).

Une image contenant texte, écriture manuscrite, dessin, encre

Description générée automatiquementVoici les Wireframes des pages « register », « values\_insertions » et « data\_table » en version mobile :

Figure 36 Wireframe page data\_table

Une image contenant texte, écriture manuscrite, Police, encre

Description générée automatiquementUne image contenant texte, écriture manuscrite, lettre, encre

Description générée automatiquement

Figure 37 Wireframe page Values\_insertion

Figure 38 Wireframe page Register

Ces wireframes me fournissent une vision globale de l'application et me permettent de mieux structurer chaque page en fonction de la hauteur et de la largeur de l'écran utilisé par l'utilisateur.

### Thème

Afin d'assurer une cohérence visuelle et une harmonie au sein de l'application, je choisis de mettre en place un thème défini.

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

Description générée automatiquementVoici la palette de couleurs que je sélectionne pour ce thème :

Figure 39 Couleur du thème de l'application

Je privilégie des teintes relativement neutres, de couleurs bleu, blanc et gris, dans le but de permettre à l'utilisateur de se concentrer pleinement sur l'analyse des informations sans être distrait par le thème. La teinte de rouge choisi est légèrement vive, afin de faire ressortir les éléments importants.

La police d'écriture choisie pour cette application est la police de Googlefont "Roboto".

« Voici un exemple de cette police d’écriture ».

C’est une police moderne, populaire et très facilement accessible. De plus, elle est conçue pour une lecture facile à l'écran, ce qui en fait une police idéale pour les interfaces utilisateur.

Une image contenant Silhouette d’animal, jouet, dessin humoristique

Description générée automatiquementJe choisi pour le logo un style graphique simpliste qui représente une image de poisson rouge, faisant référence au besoin primaire de l’application qui est le stockage et le suivi des données relatives aux minéraux contenus dans l'eau des aquariums.

Ce choix me permet d’avoir une application facilement reconnaissable, tout en rassurant l’utilisateur sur la simplicité d’utilisation de l’application.

Figure 40 Logo de l'application

### Mockups

Je réalise ensuite les mockups de chaque page, afin d’avoir des illustrations visuelles des pages, et de reproduire au mieux l'apparence visuelle attendue lors de l’implémentation.

C'est au cours de la création de ces mockups que j’applique le thème visuel précédemment défini pour l'application.

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

Description générée automatiquementUne image contenant texte, capture d’écran, nombre, reçu

Description générée automatiquementVoici les Mockups des pages « login », « values\_insertions » et « data\_table » en version mobile :

Figure 41 Mockups page Values\_insertion

Figure 42 Mockup page data\_table

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 43 Mockup page Login

Comme nous pouvons le voir sur ces illustrations, le logo, la typographie ainsi que les différentes teintes de couleurs sont alignées sur le thème choisi, assurant ainsi une cohérence visuelle entre les mockups.

Dans l’objectif de créer une application responsive à toutes les tailles d’écrans, un mockup pour chaque support de chaque page est créé, afin d'appréhender plus efficacement la manière dont les divers éléments réagissent selon la hauteur et largeur du support utilisé par l’utilisateur.

D’autres illustrations de ces mockups en version tablette et grand écran sont présentés dans l’annexe.

## Architecture

### Architecture logicielle

Afin de réaliser cette application, j’ai choisi l’architecture MVC qui signifie « Modèle Vue Controller », car c’est une architecture qui est extrêmement modulaire, et qui permet la réutilisabilité du code.

Cette architecture se compose traditionnellement de trois composants principaux que sont le Modèle, la Vue et le Contrôleur.

Cependant, j’ai décidé d’ajouter sur mon architecture le composant Routeur, qui me permet de gérer mes routes et décider des URLs affichés pour l’utilisateur.

J’ai aussi décidé d’ajouter le composant Service, afin d’encapsuler la logique métier de chaque modèle, et d’améliorer la réutilisabilité du code. De plus, il aide grandement lors de la réalisation des test unitaires.

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

Description générée automatiquementVoici une représentation complète de l’architecture de l’application :

Figure 44 Schéma de l'architecture logicielle

En premier, l’utilisateur envois une requête en direction du routeur. C’est le routeur qui, suivant l’URL de la requête et de son type, va rediriger les informations vers le contrôleur approprié.

Le contrôleur est celui qui agit comme un médiateur entre la vue, les services et les modèles.  
Il va interagir avec le service pour faciliter le passage des données vers la base de données, qui dans mon cas est une base SQL.

Ces données sont gérées de manière structurés grâce aux modèles. Il va ensuite récupérer la vue appropriée, y afficher les résultats, puis envoyer cette vue en tant que réponse à la requête de l’utilisateur.

### Arborescence des fichiers

Afin de mieux comprendre l’organisation de la hiérarchie des dossiers et fichiers en méthodologie MVC, voici une arborescence expliquant la structuration du projet.

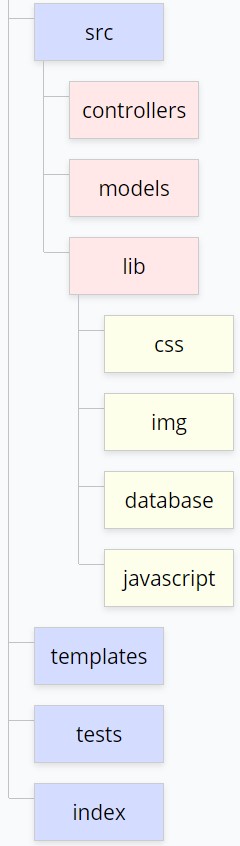
Nous avons le fichier « index », qui est le fichier par défaut lorsque l’on ouvre l’application. Il sert à réaliser une redirection vers le contrôleur approprié au démarrage de l’application.

Figure 45 Arborescence des dossiers et fichiers

Nous avons le dossier « tests », qui contient les fichiers de tests unitaires, afin de tester l’application.

Nous avons le dossier « templates », qui contient tous les templates HTML, qui servent de vues dans le modèle MVC.

Enfin, nous avons le dossier « src », qui signifie « source ». Ce dossier contient un dossier « controllers », avec les fichiers qui servent de contrôleurs à l’application. Il contient aussi le dossier « models » qui contient les différents modèles et les services en lien avec ces modèles.

Le dossier « lib » qui signifie « librairie », contient tous les fichiers annexes mais nécessaire au bon déroulement de l’application.  
Nous y retrouvons le dossier « css », qui contient les fichiers CSS afin de donner du style à nos templates et le dossier « img », ou image, qui contient les illustrations.

Le dossier « database » contient le fichier de service permettant la connexion à la base de donnée, ainsi que le fichier « .env » permettant de cacher ces informations sensibles.

Enfin, le dossier « javascript » contient les fichiers javascript permettant de rendre les templates dynamique pour l’utilisateur.

## Tests Unitaires

Pour pouvoir réaliser des tests en continus durant tout le développement de l’application, j’ai mis en place des test unitaires, avec le Framework PHPUnit. Ces tests me permettent de tester chaque fonctionnalité séparément grâce aux services et aux modèles mis en place.

Ainsi, à chaque fois que je réalise une nouvelle implémentation ou modification sur l’application, je peux faire fonctionner mes tests unitaires pour vérifier chaque fonctionnalité, afin de mettre en évidence les nouveaux bugs ou anomalies, pour pouvoir ensuite les corriger.

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

Description générée automatiquementVoici un des fichiers de tests unitaires que j’ai mis en place :

Figure 46 Fichier de tests unitaires

En indiquant la ligne de commande « **phpunit .\tests\tests\_unit.php** » dans le terminal de commande, je déclenche la méthode « setUp », qui met en place l’environnement de test en créant l’instance de connexion à la base de données, ainsi que les instances des services nécessaire aux tests.

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

Description générée automatiquementPHPUnit exécute ensuite l’ensemble de mes tests unitaires, puis me donne les résultats des tests.

Figure 47 Résultats des tests unitaires

## Cybersécurité

### Sécurisation de la base de données

Mes variables de connexion à ma base de données sont des informations sensibles qui ne doivent pas être affiché dans le code de mes contrôleurs ou mes services pour des raisons de sécurité. Je décide donc de créer un fichier d’environnement spécifique dédié à ces variables.

J’ai nommé le nom de ce fichier « .envdb.php ». Il est conçu comme un fichier caché, signalé par son point initial. De ce fait, il reste hors de la vue des explorateurs de fichiers, contribuant ainsi à atténuer les risques d'attaques. Cette approche est une mesure de cybersécurité recommandée, visant à renforcer la sécurité des données sensibles.

Figure 48 Fichier de variables de base de données

Au sein de ce fichier, j’utilise la superglobale « $\_SERVER[‘http\_HOST’ ] » du langage de programmation PHP, qui me permet de connaître l’environnement sur lequel l’application est déployé.

Ainsi, suivant si l’application est sur un environnement de développement ou un environnement de production, le fichier peut définir de lui-même les variables de base de données nécessaires, et se connecter à la base de données de l’environnement sans intervention de ma part.

Cette approche me facilite la réalisation de l'intégration continue et du déploiement continu, s'inscrivant dans une démarche de CI/CD.

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

Description générée automatiquementÉtant donné que le développement de cette application est réalisé en utilisant le principe de sauvegarde distant via Github, il existe un risque potentiel de divulgation des informations sensibles si une personne malveillante arrive à accéder au répertoire sur lequel le code est conservé. Cette personne pourrait engendrer une altération de la base de données si elle arrive à accéder aux informations de connexion à la base de données. Ainsi, en regroupant l'ensemble de ces informations sensibles au sein d'un fichier unique, je suis en mesure d'exclure ce fichier lors de la sauvegarde de mon code sur GitHub, en utilisant le fichier gitignore.

Figure 49 Fichier gitignore excluant les données sensibles

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

Description générée automatiquementLorsque l’application nécessite de l’utilisation de ces variables d’environnement de connexion à la base de données, celle-ci s’effectue au sein de la classe Database.

La connexion est conservée dans une variable privée sous en principe d’encapsulation. Ainsi, seule la méthode « get\_connection() » fournis cette connexion, ce qui renforce la sécurité de la base de données.

Figure 50 Instance Database

Les services qui utilisent cette instance « Database » fonctionnent aussi sur un principe d’encapsulation.

Cette instance « Database » est conservé en tant que variable privée dans les services pour ne pouvoir y être accessible que à l’intérieur de ces services.

Ainsi, seules les méthodes prédéfinies dans les services permettent d’accéder à cette instance « Database ».

Figure 51 Service UserRepository

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

Description générée automatiquementVoici en exemple le service « UserRepository », en lien avec le modèle « User ». Nous pouvons voir la variable privé « database » qui contient l’instance « Database », représentant la connexion à la base de données. La méthode « set\_database » permet d’initié l’instance de « Database » et d’établir la connexion. La méthode « createUser » peut ensuite utiliser cette instance dans la variable privée pour enregistrer un nouvel utilisateur dans la base de donnée sur la table « users ».

### Hachage des mots de passe utilisateurs

L’article 32.1 du Règlement général sur la protection des données (RGPD) stipule que

*« Le responsable du traitement et le sous-traitant mettent en œuvre les mesures techniques et organisationnelles appropriées afin de garantir un niveau de sécurité adapté au risque, y compris entre autres, selon les besoins :*

1. *la pseudonymisation et le chiffrement des données à caractère personnel ».* [3]

Dans le contexte de la protection des données personnelles conformément à ce Règlement, le hachage des mots de passe est une mesure technique de sécurité qui contribue à garantir un niveau de protection approprié.

En effet, en enregistrant un résultat de hachage dans la base de données à la place d’un mot de passe visible, une personne mal intentionnée ayant accès à la base de données ne pourra pas se servir de ces mots de passe pour usurper l’identité des utilisateurs sur l’application, ni sur d’autres plateformes extérieures à l’application qui pourrait représenter un impact bien plus important pour l’utilisateur.

Pour établir ce hachage de mots de passe sur mon application afin de les transformer en chaîne de caractères crypté, j’utilise la fonction de PHP « password\_hash() ».

Comme indiqué sur la documentation officielle de PHP :

*« La fonction password\_hash() crée un nouveau hachage en utilisant un algorithme de hachage fort et irréversible. »* [4]

Cette fonction utilise un algorithme « bcrypt », qui hache le mot de passe de l’utilisateur en une chaîne de soixante caractères, apparemment aléatoire, et dont il est impossible de retrouver le mot de passe initial.

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

Description générée automatiquementVoici l’implémentation de cette fonction dans l’application.

Figure 52 Hachage du mot de passe utilisateur

Durant le processus de création de compte réalisé par un utilisateur, c’est le résultat du hachage de son mot de passe réalisé par la fonction « password\_hash » qui est est donné en argument lors de la méthode d’enregistrement de l’utilisateur sur la table « users » de la base de données.

Afin que l’utilisateur puisse par la suite se connecter, il est nécessaire de savoir si le mot de passe qu’il renseigne est identique au résultat de hachage désormais enregistré dans la base.

Pour cela, la fonction PHP « password\_verify() » ne va pas décrypter le hachage, mais simplement prendre en paramètre un mot de passe, ainsi que le résultat de précédent hachage représentant ce même mot de passe.

Il va ensuite hacher le nouveau mot de passe, puis vérifier que ce résultat est identique au résultat du précédent hachage en retournant un booléen. Si le résultat du nouveau hachage est identique au résultat de l’ancien hachage, il retournera true, sinon, il retourna false.

Ainsi, a aucun moment il n’est possible de connaître le mot de passe du hachage sans avoir le mot de passe initial.

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

Description générée automatiquementVoici l’implémentation de cette fonction dans l’application.

Figure 53 Vérification d'un résultat de hachage

Le service appelé « userRepository » va essayer de récupérer l’utilisateur sur la table « user » en base de données en utilisant la méthode « getUserByEmail » qui permet de récupérer l’enregistrement en lien avec l’email donné en argument.

Si aucun enregistrement n’est récupéré, c’est que aucun enregistrement n’existe avec cet email. Une exception est donc levée, indiquant que les identifiants sont invalides.

Si un enregistrement est récupéré, une instance du modèle « user » contenant toutes les informations de l’enregistrement est généré dans la variable « $user ». On va ensuite utiliser la fonction « password\_verify() » avec le mot de passe non crypté en premier argument, et le mot de passe précédemment haché en second argument, que l’on va récupérer avec l’accesseur du modèle user « get\_password\_user() ».

Si le résultat de cette fonction est true, on peut autoriser la connexion au compte de l’utilisateur. Sinon, une exception est donc levée, indiquant que les identifiants sont invalides.

Il est important d’afficher le même message d’erreur à l’utilisateur, qu’il s’agisse d’une erreur d’email ou une erreur de mot de passe, afin qu’il n’y ai pas d’indication qui lui permettrait de récupérer des informations sensibles.

## Déploiement

### Déploiement Serveur

Afin de déployer mon application, j’ai sélectionné l’hébergeur web LWS car c’est un hébergeur français avec des serveurs en France. Cette localisation me permet d’améliorer la réactivité de l’application pour les utilisateurs français et pour ceux aux alentours.

Ils ont aussi des prix attractifs, et une capacité de disponibilité de 99,99% garantie, ce qui est suffisant pour l’application que je souhaite déployer.

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

Description générée automatiquementDans un objectif d’anticipation d’une augmentation future du nombre d'utilisateurs de l'application, il est impératif que je planifie une capacité de traitement et de calcul adéquate pour faire face à cette croissance. Pour cela, j’ai donc choisi l’offre LWS starter, qui me donne 256 Mo de mémoire, 32 Go de RAM, ainsi que 30 processus. Cette capacité de traitement me semble raisonnable pour l’utilisation actuelle et futur de l’application.

Figure 54 Tableau des offres d'hébergement LWS

Le nom de domaine m’est offert lors de l’abonnement annuel à l’hébergeur. Pour cette application, je choisi le nom de domaine [www.aquanote.fr](http://www.aquanote.fr), sur lequel le site est et sera héberger jusqu’en aout 2024 renouvelable. Je demande à l’hébergeur le certificat SSL sur ce nom de domaine afin de sécuriser la communication entre le navigateur web de l’utilisateur et le serveur web de l’application.

Afin de déployer mon application PHP sur le serveur, j’utilise le protocole File Transfer Protocol (FTP) avec le client de transfert de fichier open source « FileZilla ». Ce protocole requiert une connexion à la machine de destination, nécessitant un nom d'utilisateur et un mot de passe correspondant aux identifiants requis pour accéder à la machine. Il faut aussi choisir le port, qui par défaut est 21 pour ce type de protocole. Une fois connecté, il me suffit de transférer le dossier contenant l’application sur le serveur, et d’attendre quelques minutes pour voir apparaître mon application sur le nom de domaine sélectionné.

### Déploiement de la base de données

Je crée ensuite la base de données ainsi que les tables en lien avec mon application sur le serveur.

Pour réaliser cela, l’hébergeur possède une interface qui permet d’accéder à l’interface PHPmyAdmin du serveur. Je peux donc accéder à l’emplacement de ma base de données en indiquant le nom de compte et le mot de passe utiliseur, qui est identique à celui pour pouvoir accéder à mon compte sur l’hébergeur.

Afin de faciliter l’intégration et le déploiement continue, je décide d’éditer un fichier SQL avec le logiciel MySQLWorkbench sur lequel je crée l’ensemble de mes tables ainsi que leurs relations. J’édite aussi un autre fichier SQL contenant des fausses données utilisateurs pour ces tables.

Ainsi, à chaque intégration ou déploiement, je peux facilement recréer l’ensemble de mes tables, pour pouvoir vérifier le bon comportement de l’application.

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

Description générée automatiquementVoici un exemple du fichier SQL que j’ai édité représentant la création de la table « users », « aquariums » et « type\_analysis » :

Figure 55 Fichier SQL de création de tables

Nous pouvons y voir les clés primaires qui sont auto-incrémentés. Nous pouvons aussi voir les clés étrangères qui y sont déclarés, et sur lequel j’ajoute la contrainte de clé étrangère en lien avec la clé primaire correspondante.

L’implémentation de la suppression en cascade est mise en place sur chacune de ces clés étrangères, afin de supprimer automatiquement l’enregistrement de la table si l’enregistrement associé à cette clé primaire est supprimé.

# Conclusion

### Conclusion professionnelle

Mon expérience au cours de cette année 2022 / 2023 est extrêmement positive, que cela soit d’un point de vue personnel ou professionnel.

J’ai eu la chance d’effectuer une alternance très intéressante dans l’entreprise CELAD, ou j’ai pu découvrir le fonctionnement d’une ESN, ainsi que la vie d’entreprise au sein d’un grand groupe.

Je remercie particulièrement l’équipe CAT, avec qui j’ai travaillé durant toute mon année. C’est une équipe incroyable, avec des responsables toujours à l’écoute, toujours dans l’entraide. Merci Olivier. Merci Andréa. Ainsi que tous mes autres collègues, qui m’ont aussi toujours aidé et soutenu. J’ai vraiment été dans un cadre ou toute l’équipe a participé à me faire monter en compétence de la meilleure des manières possibles.

Même si dans ce mémoire, je présente majoritairement le projet destiné à Renault que j’ai réalisé durant la seconde moitié de mon alternance, la première moitié de mon alternance a été réalisé sur l’application Web « Shelter » en lien avec le Framework interne de l’entreprise « CAT ».

C’est sur ce projet que je suis monté en compétence en apprenant le Framework Angular, que je n’avais jamais utilisé, ou encore le serveur Flask de Python ainsi que la base de données MongoDB.

Sur cette application web « Shelter », j’ai eu comme tache d’améliorer le « Campaign Builder », qui est une fonctionnalité avec des composants Angular qui réagissent entre eux au drag and drop de l’utilisateur sur un principe de composant récursif (semblable à des dossiers contenus dans des dossiers, …).

J’ai aussi pu implémenter de manière individuel l’intégralité du principe de thème Angular sur tout le projet. Pour cela, j’ai changé les couleurs de toute l’application en implémentant la fonctionnalité de ce thème, et en ajoutant un dark thème activable par un simple bouton.

J’ai pu découvrir la méthode Agile au sein de l’entreprise avec les daily, les sprint-planning, les retro-planning, les poker-planning, le travail segmenté sous format de ticket à la tâche, et les revues de codes que j’ai aussi réalisé pour mes collègues.

J’ai trouvé cela très plaisant de travailler à plusieurs sur un même projet, en avançant ensemble vers le même objectif.

Enfin, le projet Renault, que je vous présente sur ce mémoire et que j’ai développé en commençant de zéro sous la supervision de mon responsable, m’a permis d’apprendre des choses complétement différentes et complémentaires à ma première partie d’alternance.

J’ai découvert l’ensemble de la création de projet de la conception jusqu’à la livraison au client.  
J’ai aussi participé aux réunions avec le client, à apprendre à maintenir une communication constante avec lui afin de discuter des problèmes rencontrés dans l’objectif de trouver des solutions qui répondent au mieux à ces attentes. Cela m’a fait prendre conscience de l’importance de la communication entre le client et le développeur lors de la création d’un projet.

J’ai aussi appris à fournir différents livrable en respectant le planning. J’ai quelquefois rencontré des difficultés à respecter ce planning, mais je n’ai jamais eu plus de trois jours de retard sur celui-ci, et la livraison finale a été effectué à temps.

Enfin, j’ai pris beaucoup de plaisir à voir le projet se créer sous mes yeux de jours en jours, de voir que les livrables que je fournissais étaient utilisés par l’équipe FUTE, et qu’ils me fournissaient des retours client afin de continuer à améliorer le projet.

### Conclusion personnelle

Ayant réalisé la majorité de mon temps d’apprentissage à l’école durant mon année précédente de Licence2, la décision de réaliser cette année de Licence3 en alternance a été une vraie suite logique sur mon parcours professionnel et personnel.

Cette opportunité m'a offert la possibilité de mettre en pratique l'ensemble des compétences que j'ai acquises tout au long de ma formation, et de les appliquer au sein de l'entreprise où j'ai réalisé mon alternance.

J’ai pu en parallèle poursuivre ma formation à l'école de l'IPI, consolidant ainsi mes connaissances théoriques et académiques.

J’ai eu la chance d’être très bien entouré durant toute l’année, que cela soit à l’école ou en entreprise, et je remercie chacun d’entre eux pour cette belle année.

J’ai personnellement découvert que j’apprécie beaucoup la vie de bureau, ainsi que l’ambiance générale dans le monde de l’informatique. Le fait de passer la journée à résoudre des problèmes en équipe et en collaboration, tout en construisant un projet qui évolue de semaine en semaine est très gratifiant et crée des liens entre les personnes.

Enfin, cette année m’a permis de prendre confiance en moi et en mes capacités dans le milieu de l’informatique et du développement. Désormais, j’ai le ressenti que j’ai la capacité d’être employable par une entreprise de développement. Je pense désormais que je commence à avoir les compétences pour produire des fonctionnalités, des parties d’application ou bien même une application intégrale pour un client.

### Perspective future

Ces douze derniers mois ont confirmé le fait que j’apprécie le fait d’évoluer professionnellement dans le monde du développement et je pense avoir trouvé ma voie dans ce milieu.

Cependant, j’ai découvert un intérêt particulier pour la spécification sur la BigData, grâce à des collègues qui travaillent dans ce milieu et avec qui j’ai eu la chance de pouvoir discuter plus en détail de leurs professions. Ils m’ont montré leurs réalisations ainsi que les sujets sur lesquels ils travaillent, et c’est avec un grand enthousiasme que je souhaite moi aussi me lancer dans cette spécification.

Après plusieurs recherches personnelles sur chaque profession en lien avec la spécification que représente la BigData, j’ai pris le choix de continuer ma formation sur le métier de Data-Engineer.

Je souhaite me professionnaliser en développant mes compétences en création d’ETLs, gestion de datalake, ainsi qu’en création de dashboard dans l’objectif de présenter les données de la manière la plus efficiente possible.

Ainsi, je me suis inscrit dans une école privée sur Toulouse afin de réaliser un Mastère1 et un Mastère2 spécialisé dans ce métier de Data-Engineer.

Après de très nombreuses recherches personnelles, j’ai été contacté par une ESN situé sur Toulouse qui souhaite recruter un alternant Data-Engineer. Si je suis recruté dans cette entreprise, j’aurais l’occasion de travailler avec une équipe spécialisée dans la Data avec un tuteur qui a 6 ans de spécialisation dans le métier de Data-Engineer.

A l’heure où j’écris ces lignes, tous les entretiens ont été validés avec cette ESN et nous sommes en train de faire les documents administratifs pour valider ce contrat d’alternance d’une durée de deux ans.

Une fois que ce Mastère sera terminé, je souhaite arrêter mes études et rentrer pleinement dans la vie active en tant que Data-Engineer, en espérant faire une belle et longue carrière dans cette profession.

# Bibliographie et citations

|  |  |
| --- | --- |
| [1] | V. GARDEAU, «CELAD,» Toulouse. |
| [2] | Angular.io. [En ligne]. Available: https://angular.io/guide/update-to-version-15#angular-v15-supports-nodejs-versions-1420x-1613x-and-1810x. |
| [3] | Règlement.général.sur.la.protection.des.données(RGPD). [En ligne]. Available: https://www.gdpr-expert.eu/article.html?id=32#textesofficiels. |
| [4] | Documentation.officiel.PHP. [En ligne]. Available: https://www.php.net/manual/fr/function.password-hash.php. |

# Glossaire

# Table des illustrations

[Figure 1 Chiffre d’affaires et collaborateurs 7](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366771)

[Figure 2 Interaction entre le TECU et le logiciel FUTE 8](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366772)

[Figure 3 Exemple de ligne de log GUI de type signal update 10](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366773)

[Figure 4 Parcours utilisateur de l'application 11](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366774)

[Figure 5 Composant d'affichage des logs 11](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366775)

[Figure 6 Barre de défilement du temps 11](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366776)

[Figure 7 Architecture logicielle Back-end 12](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366777)

[Figure 8 Architecture Front-end du stockage des Events 13](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366778)

[Figure 9 Architecture Front-end de l'affichage d'un Event 13](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366779)

[Figure 10 Planning de la réalisation du développement de l'application 14](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366780)

[Figure 11 Composant d'affichage des logs 16](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366781)

[Figure 12 Ligne de log GUI signifiant le déclenchement d'un nouveau test 17](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366782)

[Figure 13 Interface LogState 18](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366783)

[Figure 14 Méthode onEvent du Log Service 18](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366784)

[Figure 15 Fonctionnement de l'HTML du composant Logs 19](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366785)

[Figure 16 Fonctionnement du Typescript du Composant Log 19](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366786)

[Figure 17 Message d'erreur affiché sur le logiciel lors de l’affichage d’un test spécifique 20](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366787)

[Figure 18 Méthode onScroll du Composant Log 21](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366788)

[Figure 19 Nouvelle implémentation de l'HTML du composant Logs 22](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366789)

[Figure 20 Dockerfile Backend 24](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366790)

[Figure 21 Fichier requirement.txt 24](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366791)

[Figure 22 Dockerfile frontend 25](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366792)

[Figure 23 Dockerignore du dockerfile frontend 25](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366793)

[Figure 24 Version de production du projet angular 26](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366794)

[Figure 25 Fichier nginx.conf du dockerfile frontend 26](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366795)

[Figure 26 Docker compose 27](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366796)

[Figure 27 Résultat du docker-compose 28](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366797)

[Figure 28 Plan de navigation utilisateur 31](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366798)

[Figure 29 Diagramme des cas d'utilisation 32](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366799)

[Figure 30 Diagramme d'activité de création de compte 33](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366800)

[Figure 31 Diagramme de séquence de création de compte 34](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366801)

[Figure 32 Diagramme de classes des entités 35](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366802)

[Figure 33 Diagramme de classes des services 35](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366803)

[Figure 34 Modèle Conceptuel De Données 36](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366804)

[Figure 35 Modèle Logique de Données 37](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366805)

[Figure 36 Wireframe page data\_table 38](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366806)

[Figure 37 Wireframe page Values\_insertion 38](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366807)

[Figure 38 Wireframe page Register 38](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366808)

[Figure 39 Couleur du thème de l'application 39](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366809)

[Figure 40 Logo de l'application 39](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366810)

[Figure 41 Mockups page Values\_insertion 40](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366811)

[Figure 42 Mockup page data\_table 40](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366812)

[Figure 43 Mockup page Login 40](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366813)

[Figure 44 Schéma de l'architecture logicielle 41](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366814)

[Figure 45 Arborescence des dossiers et fichiers 42](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366815)

[Figure 46 Fichier de tests unitaires 43](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366816)

[Figure 47 Résultats des tests unitaires 43](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366817)

[Figure 48 Fichier de variables de base de données 44](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366818)

[Figure 49 Fichier gitignore excluant les données sensibles 44](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366819)

[Figure 50 Instance Database 45](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366820)

[Figure 51 Service UserRepository 45](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366821)

[Figure 52 Hachage du mot de passe utilisateur 46](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366822)

[Figure 53 Vérification d'un résultat de hachage 47](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366823)

[Figure 54 Tableau des offres d'hébergement LWS 48](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366824)

[Figure 55 Fichier SQL de création de tables 49](file:///C:\xampp\htdocs\licence3\mémoire\brouillon5_memoire.docx#_Toc143366825)

# Annexes