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. Matrice des compétences 4](#_Toc143460640)

[III. Présentation personnelle 7](#_Toc143460641)

[IV. Présentation de l’entreprise 8](#_Toc143460642)

[A. Historique 8](#_Toc143460643)

[B. Activités et Chiffres-clé 8](#_Toc143460644)

[C. Mon équipe et mes activités 9](#_Toc143460645)

[V. Application Web FUTE 10](#_Toc143460646)

[A. Présentation du client 10](#_Toc143460647)

[1. Équipe et activités 10](#_Toc143460648)

[2. Besoins exprimés 11](#_Toc143460649)

[B. Définition du projet 12](#_Toc143460650)

[1. Cahier des charges 12](#_Toc143460651)

[2. Architecture des interfaces 13](#_Toc143460652)

[3. Architecture logicielle 14](#_Toc143460653)

[C. Méthodologie du projet 16](#_Toc143460654)

[1. Planning 16](#_Toc143460655)

[2. Logiciels de travail collaboratif 17](#_Toc143460656)

[3. Méthode Agile 17](#_Toc143460657)

[4. Réunions client 17](#_Toc143460658)

[D. Implémentation du composant log et de son service 18](#_Toc143460659)

[1. Contexte et objectif 18](#_Toc143460660)

[2. Interface du composant 18](#_Toc143460661)

[3. Mise en œuvre et réalisation 19](#_Toc143460662)

[4. Problématique rencontrée et solution trouvée 22](#_Toc143460663)

[E. Transformation du projet en livrable 25](#_Toc143460664)

[1. Création d’un Dockerfile 25](#_Toc143460665)

[2. Validation du livrable par le client 31](#_Toc143460666)

[3. Maintenance du projet 31](#_Toc143460667)

[VI. Projet personnel : Outil de sauvegarde et de suivi des analyses d’eau 32](#_Toc143460668)

[A. Présentation du projet 32](#_Toc143460669)

[B. Cahier des charges 33](#_Toc143460670)

[C. Plan de navigation utilisateur 33](#_Toc143460671)

[D. Conception UML 34](#_Toc143460672)

[1. Diagramme de cas d’utilisation 34](#_Toc143460673)

[2. Diagramme d’activité 35](#_Toc143460674)

[3. Diagramme de séquence 36](#_Toc143460675)

[4. Diagramme de classe 37](#_Toc143460676)

[E. Conception MCD et MLD 38](#_Toc143460677)

[1. Modèle conceptuel de données 38](#_Toc143460678)

[2. Modèle logique de données 39](#_Toc143460679)

[F. Maquettage 40](#_Toc143460680)

[1. Wireframes 40](#_Toc143460681)

[2. Thème 41](#_Toc143460682)

[3. Mockups 42](#_Toc143460683)

[G. Architecture 43](#_Toc143460684)

[1. Architecture logicielle 43](#_Toc143460685)

[2. Arborescence des fichiers 44](#_Toc143460686)

[H. Tests Unitaires 45](#_Toc143460687)

[I. Cybersécurité 46](#_Toc143460688)

[1. Sécurisation de la base de données 46](#_Toc143460689)

[2. Hachage des mots de passe utilisateurs 48](#_Toc143460690)

[J. Déploiement 50](#_Toc143460691)

[1. Déploiement Serveur 50](#_Toc143460692)

[2. Déploiement de la base de données 51](#_Toc143460693)

[VII. Conclusion 52](#_Toc143460694)

[1. Conclusion professionnelle 52](#_Toc143460695)

[2. Conclusion personnelle 53](#_Toc143460696)

[3. Perspective future 54](#_Toc143460697)

[VIII. Bibliographie et citations 55](#_Toc143460698)

[IX. Glossaire 55](#_Toc143460699)

[X. Table des illustrations 56](#_Toc143460700)

[XI. Annexes 57](#_Toc143460701)

# Matrice des compétences

|  |  |
| --- | --- |
| **Compétences** | **Pages** |
| BC01 : Concevoir des applications numériques en intégrant les recommandations de sécurité | |
| Formaliser les procédures des services utilisateurs pour recenser les résultats attendus. | VI.D.1 Diagramme de cas d’utilisation  VI.D.2 Diagramme d’activité  VI.D.3 Diagramme de séquence |
| Prendre en compte les impératifs utilisateurs en respectant les contraintes des recommandations qualité de la norme en vigueur pour l’architecture des logiciels. | V.B.1 Cahier des charges |
| Concevoir l’architecture d’une solution fiable en identifiant les spécificités d’une activité pour produire du logiciel générique réutilisable. | V.B.3 Architecture logicielle  V.D.1 Contexte et objectif  VI.G.1 Architecture logicielle  VI.G.2 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. | VI.G.1 Architecture logicielle  VI.G.2 Arborescence des fichiers |
| Envisager toutes les possibilités, même les plus improbables pour livrer un logiciel déterminé en recherchant systématiquement l’erreur ou le dysfonctionnement. | V.D.4.a) 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. | VI.J.1 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. | VI.F.1 Wireframes  VI.F.3 Mockups |
| Identifier les risques et leur niveau de criticité pour permettre leur prévention. | VI.I.1 Sécurisation de la base de données  VI.I.2 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. | V.D.3.b) Architecture du service  V.D.3.c) Architecture du composant  VI.D.4 Diagramme de classe |
| 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. | VI.I.1 Sécurisation de la base de données  VI.I.2 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. | VI.D.1 Diagramme de cas d’utilisation  VI.D.2 Diagramme d’activité  VI.D.3 Diagramme de séquence |
| 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. | IV.C 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. | V.B.1 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. | VI.E.1 Modèle conceptuel de données  VI.E.2 Modèle logique de données |
| S’insérer dans l’urbanisation présente et future du système d’informations en concevant des éléments logiciels réutilisables et structurés en couches. | V.B.3.b) Front-end  VI.G.1 Architecture logicielle  VI.G.2 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. | V.D.1 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. | V.C.1 Planning |
| 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. | V.C.3 Méthode Agile |
| Clôturer une mission de développement en faisant valider le livrable par les parties concernées et en respectant les préconisations CFTL. | V.E.2 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. | V.C.4 Réunions client  V.D.4.b) Explication de la problématique au 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. | VI.H 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. | V.D.4.b) Explication de la problématique au 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. | V.C.3 Méthode Agile  V.C.4 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. | V.D.4.b) Explication de la problématique au 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. | V.D.3.b) Architecture du service  V.D.3.c) Architecture du composant |
| Traduire une solution algorithmique dans un langage de codage informatique avec l’utilitaire approprié. | VI.I.2 Hachage des mots de passe utilisateurs  VI.J.2 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. | V.D.4.c) 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. | V.D.4.a) Identification de la problématique  V.D.4.c) 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. | V.B.3.b) Front-end |
| Préparer des jeux d’essai en envisageant toutes les possibilités dans le but de livrer un logiciel déterminé exempt d’anomalies logiques et fonctionnelles. | VI.H 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. | V.C.1 Planning  V.C.2 Logiciels 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. | V.B.3.a) Back-end  V.B.3.b) Front-end |
| 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. | VI.E.1 Modèle conceptuel de données  VI.I.2 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. | V.B.3.a) Back-end  V.B.3.b) Front-end  V.D.3.b) 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. | V.E.1 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 faire une belle carrière en boulangerie durant plus de 10 ans, travailler dans de nombreuses boulangeries pâtisseries artisanales, ayant chacune leur manière de procéder, leurs recettes et techniques.

Durant l’année 2019, je me suis lancé dans le monde de l’entreprenariat en supplément de mon travail de boulanger. Je me suis donc déclaré Auto-entrepreneur, et j’ai commencé à réaliser différentes missions pour des clients divers ou pour des projets personnels.

C’est durant cette période que j’ai fait mon premier pas dans le monde de l’informatique en réalisant mon premier site web en utilisant Shopify et c’est avec fascination que j’ai découvert la création de site Web jusqu’à décider d’apprendre à coder sur mon temps libre mes premières pages de HTML[[1]](#footnote-1) et CSS[[2]](#footnote-2) en suivant des didacticiels.

Le point de bascule est arrivé quelques moins plus tard, où j’ai simplement discuté avec un voisin de résidence. Cette personne était tout juste diplômée jeune ingénieur informatique, et elle avait réalisé une reconversion professionnelle à partir de zéro pour en arriver là où elle en était aujourd’hui.

Elle m’a expliqué plus en détail sa profession ainsi que la journée type d’un développeur professionnel. Elle m’a aussi précisé ses missions, ses conditions de travail ainsi que l’intérêt qu’elle avait pour ce métier.

Devant la preuve de ce fait accompli, j’ai donc pris l’initiative de me lancer dans cette profession, en m’inscrivant 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 en alternance et j’ai intégré l’équipe CAT[[3]](#footnote-3), faisant partie de l’ESN[[4]](#footnote-4) CELAD.

# 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 ESN, qui signifie Entreprise de Service Numérique.

Créée à Toulouse en 1990, cette entreprise s’est principalement spécialisée 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 de son évolution, l'entreprise a étendu sa présence dans de nombreuses villes françaises.  
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 chiffre d’affaires ainsi que le nombre de collaborateurs de CELAD ne cessent d’évoluer.

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 ses 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.

Figure Chiffre d’affaires et collaborateurs

Comme le dit le Président de CELAD : « Aujourd’hui CELAD est un acteur reconnu pour sa politique sociale et pour la qualité de ses prestations. Nous continuerons la rédaction de notre Histoire sur la base de ces valeurs ! »[[5]](#footnote-5) [1]

## Mon équipe et mes activités

J’ai réalisé la totalité de mon année d’alternance avec l’équipe CAT. Il s’agit d’une équipe qui travaille sur un projet interne à CELAD.

Contrairement à la traditionnelle pratique des ESN où les clients rémunèrent des consultants, dans mon équipe, les clients paient directement pour utiliser le logiciel CAT. Cela permet à l’équipe de continuer à travailler sur le logiciel, tout en restant rentable pour l’entreprise CELAD.

Le logiciel CAT signifie « Celad Automation Tools ». Il s’agit d’un Framework de tests qui permet d’effectuer de la validation fonctionnelle complète ainsi que des tests système pour les utilisateurs.

Ce Framework est accompagné d’une Application Web appelé « Shelter ». C’est une interface graphique dans le but d’afficher les résultats des tests de CAT et de pouvoir les analyser.

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

Nous travaillons 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 et des avancés de chacun.

À la suite de ma demande et de mes interrogations sur le sujet, le responsable a mis en place un Daily durant l’année. Le Daily est une réunion quotidienne afin de pouvoir suivre l’avancer de chaque collaborateur ainsi que les différents points de blocage de chacun, dans l’objectif de s’entraider lorsque cela est nécessaire.

Durant mon année au sein de cette équipe, la plupart de mes tâches à réaliser était en lien avec l’application web Shelter, sur lequel j’ai énormément monté en compétences.

J’ai ensuite travaillé sur une mission en sous-traitance pour le client Renault, tout en restant au sein de l’équipe CAT.

# Application Web FUTE

## Présentation du client

### Équipe et activités

Au milieu de mon d’année d’alternance, notre équipe CAT a été démarchée par l’équipe FUTE Renault. Il s’agit d’une équipe de collaborateurs internes à l’ESN CELAD, mais qui travaille exclusivement en sous-traitance pour le compte de Renault.

L’équipe FUTE est spécialisée dans le développement d’un logiciel spécialement conçu pour réaliser des campagnes de tests afin de vérifier le bon fonctionnement 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 nouvelles fonctionnalités.

Afin de pouvoir mettre à jour ces véhicules, le logiciel FUTE a été créé. C’est un logiciel qui fonctionne en connexion direct avec le TECU du véhicule.

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

Description générée automatiquementTECU[[6]](#footnote-6) signifie « Transmission Electronic Control Unit ». Il s’agit d’un module électronique interne à la voiture qui est dédié à définir l’état global du véhicule afin d’optimiser les changements de vitesse et les performances globales de transmission de vitesse. Pour réaliser cela, ce module collecte l’intégralité de l’évolution des informations du véhicule.

Lorsque le logiciel FUTE se connecte au TECU, il 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 découvrir et corriger des bugs ou correctifs à améliorer, avant de mettre en production la mise à jour sur l’ensemble du parc automobile.

Figure Interaction entre le TECU et le logiciel FUTE

### Besoins exprimés

Le logiciel FUTE, à chaque réalisation de campagne de tests avec le TECU, génère les résultats de ces tests sous un format de fichier texte rangé 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 internes à Renault, il a été signalé une difficulté persistante à lire et analyser ces fichiers logs générés par le logiciel.

En effet, la génération de logs est très importante par campagne de tests, et le fait d’avoir des fichiers contenant plusieurs centaines de milliers de lignes de logs est difficilement compréhensible pour un utilisateur souhaitant comprendre le résultat d’un test dans l’objectif de l’analyser.

De plus, l’état global du véhicule, qui consiste en la compréhension de l’ensemble des informations du véhicule à un moment précis, doit pouvoir être facilement accessible 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 la lecture de 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’analyse 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 « Shelter » 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 « Shelter » reliée 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

Afin 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 lors de la validation du montant de la facturation du projet.

Durant nos interactions avec le client, les attentes ont été clairement définies et exprimées, ce qui nous a grandement aidé dans la compréhension du résultat attendu, ainsi que lors de la conception.

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 affichées sur l’application Web.

Les informations dites « pertinentes » sont au préalablement marquées au sein des lignes de logs par les caractères « GUI[[7]](#footnote-7) »,sigle qui signifie Graphical User Interface. Elles sont reformatées et nommées par type afin de faciliter la séparation et le traitement des informations.

Figure Exemple de ligne de log GUI de type signal update

Ces informations doivent ensuite être triées par l’application et affichées par type de manière dynamique. L’objectif final est de pouvoir voir à un instant donné l’état global du véhicule.

L’affichage de l’ensemble des lignes de logs doit aussi être accessible par l’application, tout en étant segmenté par test, afin que l’utilisateur puisse avoir la possibilité de disposer d’un maximum d’informations sur l’application sans jamais avoir à ouvrir le fichier de logs.

Afin que l’utilisateur puisse naviguer entre les tests, ainsi qu’à l’intérieur de chaque test, une barre de défilement de temps semblable à celle d’un lecteur vidéo permet de se déplacer dans la temporalité, et de rejouer les différents tests de la même manière qu’ils ont été exécutés.

L’application doit avoir la capacité de lire un fichier de logs qui est en cours d’édition par le logiciel FUTE, afin d’avoir un aperçu en temps réel des résultats de la campagne de tests en train d’être exécutée.

Enfin, un affichage en format de liste des fichiers de logs permet à l’utilisateur de sélectionner et rejouer la campagne de tests de son choix. Cette liste doit être mise à jour automatiquement à chaque création d’un nouveau fichier par le logiciel FUTE.

L’application doit être accessible, performante et sécurisée afin de respecter les contraintes et recommandations qualité de la norme Renault. Une attention particulière doit être faite sur la norme WCAG[[8]](#footnote-8) (Web Content Accessibility Guidelines), qui implique de concevoir une interface utilisateur qui peut être utilisée par des personnes ayant des limitations visuelles.

### Architecture des interfaces

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

Description générée automatiquementL’interface utilisateur se compose de deux pages.  
La première s’appelle « Home ». C’est la page qui va lister l’intégralité des fichiers logs disponibles afin que l’utilisateur puisse choisir la campagne de tests qu’il souhaite rejouer.

La seconde page s’appelle « Test Result ». Elle apparait une fois la campagne choisie et se compose de trois interfaces : Test, State et Signal.

Figure Parcours utilisateur de l'application

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 rangées par type. Voici un exemple de composant qui affiche des données GUI de type « vehicule state ». Il s’agit ici de données représentant l’état actuel du contact de la clé du véhicule, ainsi que son état précédent.

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

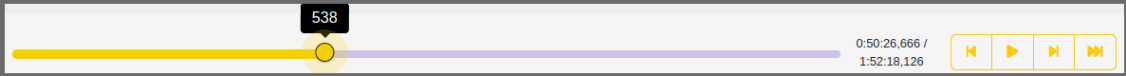
Description générée automatiquementToutes ces interfaces ont deux composants en commun.  
Le premier est la barre de défilement du temps. C’est l’outil qui permet à l’utilisateur de choisir le test et le temps précis où il souhaite afficher l’état du véhicule. Les données GUI vont par conséquent apparaître ou disparaître afin d’afficher l’état exact du véhicule à l’instant qui est sélectionné. Avec cette barre de temps, on peut donc avoir une visualisation complète de l’état global du véhicule, que cela soit au début, pendant ou à la fin d’un test.  
  
  
Le second composant qui est affiché sur toutes les interfaces est l’affichage des lignes de logs. Il affiche les logs correspondants au test qui a été en cours d’exécution au moment du temps sélectionné sur la barre de temps.

Figure Composant d'affichage des logs

Figure Barre de défilement du temps

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

### Architecture logicielle

L’architecture se compose de deux parties distinctes de l’application.

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

Description générée automatiquementPour la première partie, le Back-end, je choisis d’utiliser le langage Python.

Figure Architecture complète du logiciel

En effet, l’application nécessite que je développe un outil qui a la capacité de détecter les différents fichiers logs dans le dossier partagé.

Il nécessite aussi que je développe un second outil qui se déclenche lors de la sélection du fichier de logs. Le but de cet outil est de lire sur le fichier sélectionné chacune des lignes de log, et de séparer les informations. Cet outil détecte aussi les changements lors de l’apparition de nouvelles lignes de logs lorsque le logiciel FUTE édite le fichier.

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

Pour la création du serveur, je choisis le micro-Framework Flask afin de gérer les connexions API et Websocket avec le Front-end. Il permet une flexibilité et une simplicité de développement, idéal pour des applications web légères. De plus, il permet une intégration de grande variété de bibliothèques Python, ce qui permet au projet de facilement pouvoir prendre de l’envergure suivant les besoins futurs de l’équipe FUTE.

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 de ces éléments apparaissent jusqu’à trois fois entre les différentes interfaces. Angular propose aussi des fonctionnalités intégrées pour gérer les API Rest et les Websockets, ce qui me permet un envoi d’informations simples et sécurisées avec le Back-end.

#### Back-end

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

Description générée automatiquementVoici un exemple de fonctionnement de l’architecture logicielle Back-end lors de la lecture d’une ligne de log GUI afin de créer un objet « Event » qui est ensuite fourni au Front-end.

Figure Architecture logicielle Back-end

En premier, l’outil « Logparser » segmente la ligne de log.

Cet outil fonctionne dans un fil (Thread) afin de continuer à recevoir des informations du Websocket entre le front-end et le back-end durant le processus.

A la suite de ce parsing, il est possible de définir le type de cette ligne en lisant le cinquième élément de la ligne de log. Dans l’exemple présenté, il s’agit d’un type « SIGNAL\_UPDATE ».

Le logparser construit ensuite un objet « Event ». Il assigne dans cette Event la valeur du « type », ainsi que la valeur « sequence number »qui correspond à un compte de l’apparition des lignes de logs GUI dans le fichier afin de connaître sur le front-end l’ordre d’apparition.

Suivant le type de cette ligne GUI, des informations différentes vont être assignées à la clé « data » sous un format de relation clé-valeur, afin de rendre disponible au front-end les informations les plus pertinentes pour chacune de ces lignes.

Dans l’exemple présenté, il s’agit de l’information « name » ainsi que « value »

Une fois que cet Event est généré, il est transmis au Websocket en utilisant un mécanisme de rappel (callback). Ce rappel déclenche l'envoi de l'Event en Websocket vers le Front-end en utilisant la méthode "send".

#### Front-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 d’un Event au préalablement envoyé par le Back-end Python.

L’Event est en premier réceptionné par le canal de communication Websocket.

Suivant la valeur du type du GUI, l’information est ensuite transmise vers le service approprié. Dans notre exemple, l’Event est de type « signal\_update », il est donc envoyé vers le service en charge du stockage de ce type précis appelé « signal\_update\_service ».  
L’information est ensuite traitée et stockée dans le service en que « state », soit un état.

Figure Architecture Front-end du stockage des Events

Un état correspond à l’ensemble de toutes les informations d’un type à un instant donné. Le nouvel état qui vient d’être créé a donc toutes les valeurs des précédentes Events de type « signal update » reçus, ainsi que la nouvelle valeur du nouvel Event.

Cet état est ensuite stocké dans la variable de liste des états au sein du service.

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

Description générée automatiquementLorsque l’utilisateur souhaite afficher un état global du véhicule à un instant précis, il va sélectionner une valeur en utilisant le composant de barre de défilement du temps appelé dans le schéma « Progress Bar Component».

Cette valeur est ensuite transmise au service « Progress bar Service », qui va la partager à tous les autres services contenant des états par un principe d’abonnement. Ainsi, tous ces autres services vont fournir à leur composant respectif l’état approprié correspondant aux informations du véhicule à l’instant précis sélectionner sur la barre de défilement du temps.

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

Ce type d’architecture de stockage des données au sein des services permet de ne pas devoir récupérer à chaque fois dans le fichier log lors de la sélection d’un instant précis par l’utilisateur avec la barre de défilement du temps. De plus, il permet de ne pas recalculer l’état pour chaque composant, ce qui améliore la vitesse d’utilisation, mais contribue aussi à la politique RSE[[9]](#footnote-9) (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, je réalise un planning pour la réalisation de l’application, mais aussi afin de budgétiser le développement de celle-ci.

Figure Planning de la réalisation du développement de l'application

Le fait de réaliser une planification 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 j’ai défini les différentes tâches, estimé mon temps de travail sur chacune d’elle, et pris en compte mes disponibilités entre le centre de formation et l’entreprise.

La budgétisation de cette application étant facturée à la tâche, il est donc primordial que j’estime convenablement les différents temps de travail, et de tenir les délais, afin de ne pas être hors budget.

C’est à la fin de chaque tâche que je livre le code directement sur le répertoire Gitlab de Renault, faisant preuve du travail exécuté déclenchant la facturation.

### Logiciels de travail collaboratif

Nous avons mis en place des tickets sur le logiciel Jira 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.

Jira est un logiciel de gestion de projet. Il est principalement utilisé par les équipes de développement de logiciels, les équipes 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, j’ai la 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 me permet de faciliter la collaboration et la communication avec eux lors de problèmes techniques demandant leur intervention.

### Méthode Agile

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

Pour cela, la plateforme de gestion de dépôt Gitlab me permet de faire relire mon code par mon responsable, puis par un responsable interne à l’équipe FUTE, ce qui favorise une approche collaborative et itérative du développement.

Le partage fréquent de captures d’écran et de vidéos de l’application sur les tickets Jira me permet de présenter l’avancement du projet, afin d’avoir un retour constant du client, ce qui consolide cette méthode agile.

De plus, le travail en méthode Scrum condensé en sprint de 15 jours, ainsi que les Dailys où j’explique mon avancée ainsi que les points de blocages à mon responsable ont grandement amélioré les conditions de développement de cette application.

### Réunions client

Afin d’avoir des retours directs avec le client, mais aussi pour présenter plus en détail l’avancée du projet, trois réunions client sont planifiées.

En début de projet, la première réunion sert à définir les attentes du client, ainsi qu’à présenter le cahier des charges que nous avons conçu dans le but de faire valider la conception de celui-ci.

La seconde réunion, au milieu du projet, est planifiée juste après la livraison de l’application sous format de container docker. L’objectif est de récupérer le maximum de retours client, dans l’objectif de pouvoir réaliser les premières modifications suivant les retours client.

La dernière réunion client est une présentation de l’application finalisée à l’intégralité de l’équipe FUTE. 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 client non planifiées ont lieu lors de points de blocage importants 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 affichées de manière dynamique afin d’avoir l’état du véhicule à l’instant donné suivant la barre de défilement du temps.  
Cependant, dans certains cas d’utilisation de l’utilisateur, 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 ».

Le fait de rendre cette information disponible directement sur l’application permet donc à l’utilisateur d’éviter d’ouvrir des fichiers logs qui contiennent souvent plusieurs centaines de milliers de lignes pour récupérer ces informations.

Ce composant est disponible sur les trois interfaces de la page « Test Result ». Cela permet d’avoir un aperçu de ce composant peu importe où l’utilisateur est situé sur l’application lorsqu’un fichier est en cours de lecture.

Le service « Log Service » est spécialement dédié au composant « Log Component ». Il va contenir l’intégralité des lignes de logs. Ces données sont transmises lors du parsing du fichier et stockées de manière efficiente afin de minimiser au mieux le temps de chargement lorsque l’utilisateur réalise un changement d’affichage avec la barre de défilement du temps.

### Interface du composant

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

Description générée automatiquementL’interface de ce composant est conçu de manière la plus simple possible afin d’éviter les potentielles latences causées par un grand nombre de lignes de logs à afficher.

Figure Composant d'affichage des logs

Il consiste en une partie scrollable qui permet d’accéder à l’intégralité des informations sans prendre trop de place sur les interfaces utilisateurs.  
L’affichage des lignes de logs est réalisée sur le composant de la même manière que dans les fichiers de logs, afin de faciliter la lecture et de garder les habitudes actuelles 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 que la récupération de la donnée est effectuée lors du parsing de chaque ligne de log.

Afin de réaliser ce parsing, l’outil « LogParser » est utilisé. C’est lui qui va récupérer le nom du fichier à lire, qui va segmenter les informations contenues dans chaque ligne de log du fichier sélectionné pour transformer l’information en format d’objet « Event » transférable par protocole de communication Websocket.

Dans le contexte de l’envoi de l’intégralité des lignes de log nécessaire à l’affichage du composant « Log Composant », l’outil LogParser récupère la ligne de log en cours d’itération par l’outil, puis réalise la méthode « remove\_extra\_space » afin de retirer les espaces inutiles. Cette ligne de log est ensuite insérée dans une variable de typage tableau qui contient les autres lignes de logs.

Afin d’envoyer les informations au front-end lorsque les lignes de logs passent à un test suivant, l’équipe FUTE a créé un log GUI de type « test status » et de valeur « Running », signifiant qu’un nouveau test est en train d’être lancé.

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

Lorsque cette ligne de log apparait, ou lorsque la quantité de 250 lignes de log est franchie, le Logparser transforme la totalité des lignes de logs du tableau en un string unique, appelé aussi un « bloc ». Afin de conserver les retours à la ligne, le caractère « \n » est inséré entre chaque ligne lors de la création du bloc.

Un objet « Event » est ensuite créé, et va contenir le bloc de ligne de code, ainsi que le « sequence number », qui représente le compte de GUI depuis le début de la lecture du fichier.

L’envoi de cet objet est ensuite 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 une information du Websocket qui demanderait d’arrêter le parsing d’un fichier, même si le LogParser est en cours de fonctionnement. Comme le langage Python fonctionne nativement de manière synchrone, il est nécessaire d’utiliser un thread afin de traiter de manière parallèle ces deux exécutions simultanées.

Le fait de limiter le seuil de déclenchement d’envoi à 250 lignes de logs maximum permet d’éviter d’avoir des envois trop volumineux en Websocket, ce qui serait susceptible de créer des ralentissements pour l’utilisateur lorsqu’il navigue sur l’application

#### 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 envoyés par le Back-end.

A la réception d’un Event, le service lit l’attribut « type » de l’objet, afin de connaître le contenu de l’information et le transmet à son service approprié. Cet attribut est défini 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.

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

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

Description générée automatiquementDans le Log Service, la méthode « onEvent » reçoit et traite l’information reçue en argument.  
Cette méthode vérifie en premier la valeur de l’attribut « is new test » de l’Event.  
S’il s’agit d’un nouveau test, le Log Service génère un nouvel objet « LogState ».

Figure Interface LogState

Figure Méthode onEvent du Log Service

Ce nouvel objet créé est ensuite stocké dans le tableau de la variable « complete log states ».

S’il ne s’agit pas d’un nouveau test, le bloc de logs est inséré 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 qui représente les lignes de logs d’un seul test.

Afin de pouvoir fournir au composant les logs qui doivent être affichés, le service contient une variable « current log state » de typage « BehaviorSubjet ». Lorsque l’utilisateur sélectionne un instant à afficher dans le composant de la barre de défilement du temps, une valeur est fournie au LogService, et la méthode « ManageCurrentState » récupère dans la variable « complete log states » l’objet LogState ayant le même séquence number que la valeur sélectionnée par la barre de défilement du temps. Cet objet est ensuite assigné à la variable « current log state » pour être affiché sur le composant.

#### Architecture du composant

Le 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.

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

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

Description générée automatiquementDans le constructeur de la classe LogsComponent, le logService est assigné dans une variable pour pouvoir établir une communication entre le composant et son service.

Figure Fonctionnement du Typescript du Composant Log

Figure Fonctionnement de l'HTML du composant Logs

La méthode OnInit est automatiquement exécutée 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.

Suite au fait que la variable « current log state » est un behaviorSubjet, le composant est alerté 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 sur le composant en même temps que dans le service.

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

### Problématique rencontrée et solution trouvée

#### Identification de la problématique

A 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 corrigés.

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

Description générée automatiquementJe 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 la quantité des lignes de logs contenues au sein de ce test est de 500 000 lignes, ce qui est anormalement élevé comparé aux autres tests mis à ma disposition.

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

Après plusieurs analyses et tests fonctionnels, l’application arrive à gérer sans difficulté ni ralentissement cette quantité de lignes de logs lors du parsing et du traitement d’informations sur le back-end. Elle la gère aussi durant l’envoi et à la réception Websocket sur le front-end, ainsi que le traitement par le LogService et le 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.

Il est clairement spécifié dans le cahier des charges que ce composant doit afficher l’ensemble des lignes de logs d’un test.

Ne pouvant prendre l’initiative de modifier cette fonctionnalité sans l’accord du client, j’ai convenu avec mon responsable de fixer un rendez-vous par visioconférence avec le 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.

#### Explication de la problématique au client

Suite 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 rencontrée.

Cette réactivité me permet de ne pas prendre de retard sur le planning attendu, tout en continuant à travailler dans de bonnes conditions sur des attentes clients bien définies.

Lors de cette réunion, j’explique oralement la problématique rencontrée, en essayant d’adapter mon discours au maximum dû au fait 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 unique chargement, mais réalisera du « lazy loading », soit un chargement différé.

L’affichage des lignes de logs 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 prises durant cet entretien en modifiant mon code existant.

Afin de connaître le niveau de la barre de progression du composant, j’utilise dans la balise <section> la méthode « (scroll) » sur le fichier Html du composant.

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

Description générée automatiquementAinsi, 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.

Figure Méthode onScroll du Composant Log

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 affichée,

« scrollTop » qui contient la hauteur entre le haut de la zone scrollable total et le haut de la zone affichée, permettant de connaître la taille de toute la partie cachée car déjà scrollée.

Si la taille de la partie cachée déjà scrollée additionnée à la hauteur de la zone affichée est égale à la taille de la zone totale de la zone scrollable, alors nous sommes à la fin de la zone scrollable. Il faut donc afficher le bloc de logs suivant en plus de celui déjà affiché.

Pour cela, une nouvelle variable, appelé « log\_blocks\_displayed » est incrémentée et compte les blocs de lignes de logs à afficher.

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

Description générée automatiquement  
Désormais, sur la page Html, il m’est possible de faire apparaître les lignes de logs en « lazy loading » en utilisant cette nouvelle variable ainsi que l’index des blocs de logs à afficher. 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 Nouvelle implémentation de l'HTML du composant Logs

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

Une fois que tout est opérationnel et qu’il n’y a plus de bugs, 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 merge-request est acceptée par le responsable de l’équipe FUTE.

## 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 est réalisée à la fin de la première moitié de l’implémentation du projet. Elle 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 cette livraison finale qui sera déployée 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 le dockerfile, je construis une image docker. Il s’agit d’un modèle exécutable statique contenant tout le code, les bibliothèques ainsi que toutes les dépendances nécessaires.

Cette image sert par la suite à déployer mon container docker, semblable à 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 future de l'application.

Lorsque plusieurs dockerfiles sont déployés ensemble, comme c’est mon cas ici, la gestion de ce déploiement doit s’effectuer avec un docker-compose. C’est un fichier qui permet de gérer les applications multi-containers, et qui permet la spécification de services, réseaux et volumes spécifiques 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 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 ce dossier 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 servant à faire fonctionner l’application backend. C’est un fichier que j’ai mis à jour durant toute la durée du développement.

Figure 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 plutôt d’une documentation visant à indiquer aux futurs développeurs 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 réalisée 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 créée pour construire mon docker, et le faire fonctionner.

Une fois 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 dédié 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, qui sert à « builder », soit construire l’application, et le second est un environnement qui sert à le déployer.  
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.[[10]](#footnote-10) [2]

Figure 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 le dossier 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 de base d’un projet Angular.

Figure 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 déployée. Cette version compilée est ensuite contenue dans le dossier « app/dist/fdmtfrontend ».

Voici en exemple sur l’image ci-joint une version compilée que j’ai réalisé en local.

Figure Version de production du projet angular

Mon application est désormais en version compilée, je passe donc à l’environnement de déploiement. Pour que la partie frontend du projet soit disponible en requête HTTP[[11]](#footnote-11), 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.

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

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

Figure Fichier nginx.conf du dockerfile frontend

Le fichier nginx.conf est un fichier de configuration du serveur nginx que j’ai configuré pour être adapté à mon projet. J’indique sur ce fichier 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, 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 où mon application sera accessible.

Le dockerfile de la partie frontend terminé, je peux désormais construire mon image avec la commande « docker build -t imagefdmtfrontend . ».

Je peux ensuite construire mon container avec la commande « docker run -d -p 8080:80 --name fdmtfrontend imagefdmtfrontend », où je spécifie que le port 8080 est l’emplacement de mon application en fonctionnement.

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 testés.

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

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

Figure 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 s’appelle 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 utilisé pour la création de ce container se situe dans le dossier fdmtbackend avec la comande « build: ./fdmtbackend ».

Je connecte ensuite le port 5001 du serveur où est 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 où est déployé 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 est utilisé pour la création de ce container est situé dans le dossier fdmtfrontend avec la commande « build: ./fdmtfrontend ».

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

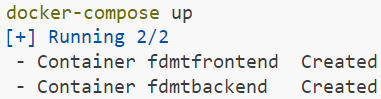
Les deux services maintenant créés dans le docker compose, j’exécute ce fichier avec la commande « docker-compose up » afin de vérifier que tout fonctionne correctement.

Figure 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 effectuée sous format de container docker, en deux étapes différentes.

La première livraison en format de container docker est réalisée au milieu du développement de ce projet. L’objectif de cette livraison 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 est réalisée à la fin du projet. Juste avant cette livraison, j’effectue 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’effectue ensuite cette seconde livraison, avec comme objectif de faire tester l’application aux utilisateurs afin de rechercher des éventuels bugs et anomalies qui seraient 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 comme 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 est réalisée. Cela signifie que la facturation regroupe l’ensemble du travail effectué de la conception jusqu’à la livraison finale.

Cependant, une fois cette livraison terminée et validée 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 réalisée par l’équipe FUTE.

L’équipe CAT reste disponible pour toute nouvelle implémentation si nécessaire avec une nouvelle facturation, et peut ainsi faire profiter de l’expérience acquise sur le projet pour faciliter le développement de nouvelles fonctionnalités.

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

## Présentation du projet

Durant mon année de licence 2, 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 conçois 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 zone de stockage d’informations pour chacun de ces aquariums si celui-ci en possède plusieurs, et la possibilité de rajouter ou supprimer des types de minéraux pour chaque aquarium sur lequel l’utilisateur souhaite réaliser des analyses.

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

L’objectif final est de donner à l’utilisateur la possibilité d’avoir une vision globale des informations sur le long terme, et de pouvoir faire parler ses 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 je sélectionne pour cette réalisation sont l’HTML, le CSS et le Javascript pour la partie Front-end de l’application.

Pour la partie Back-end, je sélectionne le langage PHP, dans le but de faciliter le déploiement tout en diminuant les frais de serveur qui seront financés personnellement.

La base de données est de type relationnelle et réalisée en MySQL.

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

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

Enfin, le logiciel est 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 Plan de navigation utilisateur

Une fois sur l’application, l’utilisateur arrive sur la page « homepage » ce 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, je réalise des diagrammes UML[[12]](#footnote-12) afin de bien me représenter les fonctionnalités de l’application et faciliter 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’utilisations 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 disponibles pour le Visiteur.

Figure 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 me 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 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 Diagramme d'activité de création de compte

Comme nous pouvons le voir dans ce diagramme, le « Accept signal » est déclenché 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 que l’email n’existe pas déjà en base de données. Il hache 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.

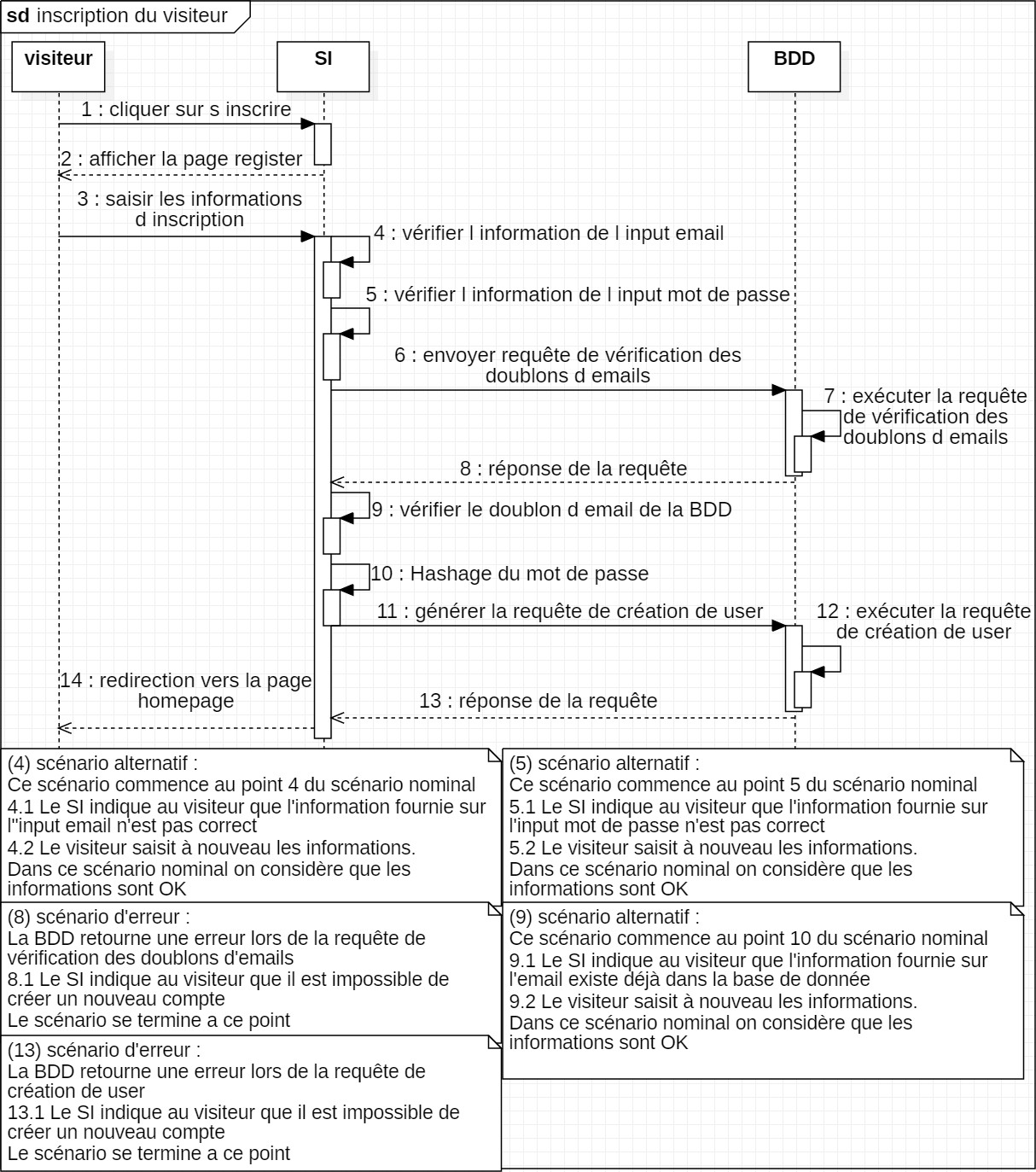
Le diagramme ci-dessous représente lui aussi le cas d’utilisation de création de compte. Il contient les trois mêmes intervenants que sur le diagramme d’activité, avec pour chacun d’eux un « Lifeline », ou ligne de vie.

Figure 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 représentent les scénarios non bloquants, ainsi que les scénarios d’erreurs, qui représentent les scénarios bloquants, sont affichés sous forme de notes séparées 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 du scénario.

### Diagramme de classe

Le diagramme de classe représente la façon dont les données sont gérées et comment il est possible d’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 représentent pour chacune leur table respective en base de données.

Figure Diagramme de classes des entités

Chacune de ces classes contient des attributs privés, ainsi que leurs accesseurs et mutateurs qui permet d’accéder à ces attributs.

La nomenclature du nom de ces classes est en pascalcase, tandis que ces attributs, ces accesseurs et ces mutateurs sont en snakecase.

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

Les instances de 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 chacune des classes.

Figure Diagramme de classes des services

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

## Conception MCD et MLD

### Modèle conceptuel de données

Le MCD[[13]](#footnote-13) (Modèle Conceptuel de Données) est une modélisation de la base de données de l’application, afin 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 lien avec les diagrammes de cas d’utilisation, d’activités et de séquences

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

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

Figure 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 permettent 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 haché 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 représentée 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 » est obligatoirement en lien qu’avec un seul enregistrement de la table « users ».  
  
Afin de faciliter le développement et la compréhension, je définis une nomenclature stricte pour le nommage des tables, qui sont en snakecase et à l’infinitif, ainsi que pour les attributs, qui sont aussi en snakecase, avec le nom de la valeur suivi du nom de la table.

### Modèle logique de données

Le MLD[[14]](#footnote-14) (Modèle Logique de Données) est une modélisation de la base de données orientée pour faciliter le développement. Son objectif est de modéliser 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 n’a pas de nécessité de créer de relations many to many. Il n’y a donc pas de table d’association dans ce modèle logique de données.

Figure Modèle Logique de Données

Durant la réalisation, je choisis volontairement de créer des relations qui réalisent 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 soient aussi supprimés, car ils n’ont plus d’utilité 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

Afin d’obtenir une représentation graphique de l’ensemble de l’application, je réalise différents maquettages. Pour leurs réalisations, je privilégie le format mobile en premier, 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 travaille de manière rapide et efficace, et que je réalise plusieurs brouillons afin d’optimiser la navigation et l’UX[[15]](#footnote-15) (User Expérience).

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 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 Wireframe page Values\_insertion

Figure 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 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 choisie 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 utilisateurs.

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

Description générée automatiquementJe choisis 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 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 Mockups page Values\_insertion

Figure Mockup page data\_table

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure Mockup page Login

Comme nous pouvons le voir sur ces illustrations, le logo, la typographie, ainsi que les différentes teintes de couleurs sont alignés 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

Pour réaliser cette application, j’ai choisi l’architecture MVC[[16]](#footnote-16) 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 de 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 Schéma de l'architecture logicielle

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

Le contrôleur est celui qui agit comme un médiateur entre la vue, les services et les modèles.  
Il interagit 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 structurée grâce aux modèles. Il récupère ensuite la vue appropriée, y affiche les résultats, puis envoie 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.

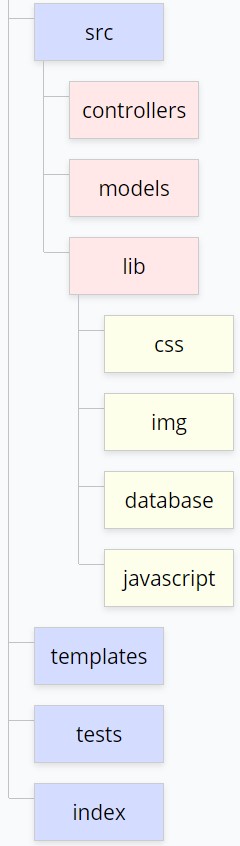
Le projet contient le fichier « index », qui est le fichier affiché par défaut lorsque l’on ouvre l’application. Il sert à réaliser une redirection vers le contrôleur approprié au démarrage de l’application.

Figure Arborescence des dossiers et fichiers

Le dossier « tests », contient les fichiers de tests unitaires, afin de tester l’application.

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

Le dossier « src » signifie « source ». Il 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 » signifie « librairie ». Il contient tous les fichiers annexes mais nécessaires au bon déroulement de l’application.

Le dossier « css » contient les fichiers CSS afin de donner du style aux templates et le dossier « img », ou image, contient les illustrations.

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

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

## Tests Unitaires

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

Ainsi, à chaque fois que je réalise une nouvelle implémentation ou modification sur l’application, je peux exécuter 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 je mets en place :

Figure 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 tests en créant l’instance de connexion à la base de données, ainsi que les instances des services nécessaires 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 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 affichées 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.

Je nomme 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 Fichier de variables de base de données

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

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[[17]](#footnote-17).

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

Description générée automatiquementDu fait 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 Fichier gitignore excluant les données sensibles

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

Description générée automatiquementLorsque l’application nécessite la connexion à la base de données, celle-ci s’effectue au sein de la classe Database, où l’utilisation des variables d’environnements sont utilisés.

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

Figure Instance Database

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

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

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

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

Figure Service UserRepository

Voici en exemple le service « UserRepository », en lien avec le modèle « User ». Nous pouvons voir la variable privée « database » qui contient l’instance « Database », représentant la connexion à la base de données.

La méthode « set\_database » permet d’initier 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 données sur la table « users ».

### Hachage des mots de passe utilisateurs

L’article 32.1 du RGPD[[18]](#footnote-18) (Règlement général sur la protection des données) 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 ».[[19]](#footnote-19)* [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 cryptés, 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. »[[20]](#footnote-20)* [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 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 fourni en argument dans 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 le mot de passe à tester, ainsi que le résultat du précédent hachage de 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, à 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 Vérification d'un résultat de hachage

Dans cette implémentation, le service « 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 en base de données. Une exception est donc levée, indiquant que les identifiants sont invalides.

Si un enregistrement est récupéré, une instance du modèle « user » qui contient toutes les informations de l’enregistrement est généré dans la variable « $user ». Je vais 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 je récupére avec l’accesseur du modèle user « get\_password\_user() ».

Si le résultat de cette fonction est true, le logiciel autorise la connexion au compte de l’utilisateur. Sinon, une exception est 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, je choisis 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 future de l’application.

Figure Tableau des offres d'hébergement LWS

Le nom de domaine m’est offert durant le processus d’abonnement annuel à l’hébergeur. Pour cette application, je choisis le nom de domaine [www.aquanote.fr](http://www.aquanote.fr), sur lequel le site est héberger jusqu’en aout 2024 renouvelable.

Je demande à l’hébergeur le certificat SSL[[21]](#footnote-21) 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 FTP[[22]](#footnote-22) (File Transfer Protocol) avec le client de transfert de fichier open source « FileZilla ». Ce protocole requiert une connexion à la machine de destination, qui nécessite 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 et qui représente la création de la table « users », « aquariums » et « type\_analysis » :

Figure Fichier SQL de création de tables

Nous pouvons voir les clés primaires qui sont auto-incrémentées. Nous pouvons aussi voir les clés étrangères qui sont déclarées, 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 ce soit d’un point de vue personnel ou professionnel.

J’ai eu la chance d’effectuer une alternance très intéressante dans l’entreprise CELAD, où 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 où 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 j’ai énormément 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 tâche 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 individuelle 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 eu la chance de découvrir la méthode Agile au sein de l’entreprise avec les daily, les sprint-planning, les retro-planning, les poker-planning. J’ai aussi découvert le travail segmenté sous format de ticket à la tâche, et les revues de codes que j’ai 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 dans ce mémoire et que j’ai développé en commençant de zéro sous la supervision de mon responsable, m’a permis d’apprendre des compétences complétement différentes et complémentaires à ma première partie d’alternance.

J’ai découvert toute la partie de création de projet de la conception jusqu’à la livraison finale 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 livrables 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 que je continue 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 que je commence à avoir les compétences pour produire des fonctionnalités, des parties d’application ou bien même des applications intégrales pour des clients.

### 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’ETL[[23]](#footnote-23) (Extract, Transform, Load), en 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 à 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 située à 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,» 2023. [En ligne]. Available: https://www.celad.com/societe/societe/. |
| [2] | Angular.io, 2023. [En ligne]. Available: https://angular.io/guide/update-to-version-15#breaking-changes-in-angular-v15. |
| [3] | Règlement.général.sur.la.protection.des.données(RGPD), 2023. [En ligne]. Available: https://www.gdpr-expert.eu/article.html?id=32. |
| [4] | Documentation.officiel.PHP, 2023. [En ligne]. Available: https://www.php.net/manual/fr/function.password-hash.php. |

# Glossaire

CAT

Celad Automation Tools **7**, **9**, **10**, **11**, **31**, **52**

CI/CD

Continuous Integration, Continuous Deployment **46**

CSS

Cascading Style Sheets **7**, **21**, **44**

ESN

Entreprise de Services du Numérique **7**, **8**, **9**, **10**, **52**, **54**

ETL

Extract, Transform, Load **54**

FTP

File Transfer Protocol **50**

GUI

Graphical User Interface **12**, **13**, **14**, **15**, **19**, **56**

HTML

HyperText Markup Language **7**, **21**, **44**, **56**

HTTP

Hypertext Transfer Protocol **28**

MCD

Modèle Conceptuel de Données **3**, **38**, **39**

MLD

Modèle Logique de Données **3**, **38**, **39**

MVC

Modèle Vue Controller **43**, **44**

RGPD

Règlement Général sur la Protection des Données **6**, **48**, **55**

RSE

Responsabilité Sociétale des Entreprises **6**, **16**

SSL

Secure Sockets Layer **50**

TECU

Transmission Electronic Control Unit **10**, **56**

UML

Unified Modeling Language **2**, **34**

UX

User Expérience **40**

WCAG

Web Content Accessibility Guidelines **12**

# Table des illustrations

[Figure 1 Chiffre d’affaires et collaborateurs 8](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525096)

[Figure 2 Interaction entre le TECU et le logiciel FUTE 10](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525097)

[Figure 3 Exemple de ligne de log GUI de type signal update 12](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525098)

[Figure 4 Parcours utilisateur de l'application 13](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525099)

[Figure 5 Composant d'affichage des logs 13](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525100)

[Figure 6 Barre de défilement du temps 13](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525101)

[Figure 7 Architecture logicielle Back-end 15](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525102)

[Figure 8 Architecture Front-end du stockage des Events 16](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525103)

[Figure 9 Architecture Front-end de l'affichage d'un Event 16](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525104)

[Figure 10 Planning de la réalisation du développement de l'application 17](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525105)

[Figure 11 Composant d'affichage des logs 19](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525106)

[Figure 12 Ligne de log GUI signifiant le déclenchement d'un nouveau test 20](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525107)

[Figure 14 Interface LogState 21](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525108)

[Figure 13 Méthode onEvent du Log Service 21](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525109)

[Figure 16 Fonctionnement du Typescript du Composant Log 22](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525110)

[Figure 15 Fonctionnement de l'HTML du composant Logs 22](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525111)

[Figure 17 Message d'erreur affiché sur le logiciel lors de l’affichage d’un test spécifique 23](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525112)

[Figure 18 Méthode onScroll du Composant Log 24](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525113)

[Figure 19 Nouvelle implémentation de l'HTML du composant Logs 25](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525114)

[Figure 20 Dockerfile Backend 27](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525115)

[Figure 21 Fichier requirement.txt 27](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525116)

[Figure 22 Dockerfile frontend 28](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525117)

[Figure 23 Dockerignore du dockerfile frontend 28](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525118)

[Figure 24 Version de production du projet angular 29](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525119)

[Figure 25 Fichier nginx.conf du dockerfile frontend 29](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525120)

[Figure 26 Docker compose 30](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525121)

[Figure 27 Résultat du docker-compose 31](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525122)

[Figure 28 Plan de navigation utilisateur 34](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525123)

[Figure 29 Diagramme des cas d'utilisation 35](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525124)

[Figure 30 Diagramme d'activité de création de compte 36](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525125)

[Figure 31 Diagramme de séquence de création de compte 37](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525126)

[Figure 32 Diagramme de classes des entités 38](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525127)

[Figure 33 Diagramme de classes des services 38](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525128)

[Figure 34 Modèle Conceptuel De Données 39](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525129)

[Figure 35 Modèle Logique de Données 40](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525130)

[Figure 36 Wireframe page data\_table 41](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525131)

[Figure 37 Wireframe page Values\_insertion 41](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525132)

[Figure 38 Wireframe page Register 41](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525133)

[Figure 39 Couleur du thème de l'application 42](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525134)

[Figure 40 Logo de l'application 42](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525135)

[Figure 41 Mockups page Values\_insertion 43](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525136)

[Figure 42 Mockup page data\_table 43](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525137)

[Figure 43 Mockup page Login 43](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525138)

[Figure 44 Schéma de l'architecture logicielle 44](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525139)

[Figure 45 Arborescence des dossiers et fichiers 45](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525140)

[Figure 46 Fichier de tests unitaires 46](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525141)

[Figure 47 Résultats des tests unitaires 46](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525142)

[Figure 48 Fichier de variables de base de données 47](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525143)

[Figure 49 Fichier gitignore excluant les données sensibles 47](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525144)

[Figure 50 Instance Database 48](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525145)

[Figure 51 Service UserRepository 48](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525146)

[Figure 52 Hachage du mot de passe utilisateur 49](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525147)

[Figure 53 Vérification d'un résultat de hachage 50](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525148)

[Figure 54 Tableau des offres d'hébergement LWS 51](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525149)

[Figure 55 Fichier SQL de création de tables 52](file:///C:\xampp\htdocs\licence3\mémoire\brouillon7_memoire.docx#_Toc143525150)

# Annexes

1. HTML : HyperText Markup Language [↑](#footnote-ref-1)
2. CSS : Cascading Style Sheets [↑](#footnote-ref-2)
3. CAT : Celad Automation Tools [↑](#footnote-ref-3)
4. ESN : Entreprise de Services du Numérique [↑](#footnote-ref-4)
5. Citation Vincent Gardeau, Président CELAD, 2023, <https://www.celad.com/societe/societe/> [↑](#footnote-ref-5)
6. TECU : Transmission Electronic Control Unit [↑](#footnote-ref-6)
7. GUI : Graphical User Interface [↑](#footnote-ref-7)
8. WCAG : Web Content Accessibility Guidelines [↑](#footnote-ref-8)
9. RSE : Responsabilité Sociétale des Entreprises [↑](#footnote-ref-9)
10. Site de documentation Angular.Io, « Breaking change in Angular V15”, 2023, <https://angular.io/guide/update-to-version-15#breaking-changes-in-angular-v15> [↑](#footnote-ref-10)
11. HTTP : HyperText Transfer Protocol [↑](#footnote-ref-11)
12. UML : Unified Modeling Language [↑](#footnote-ref-12)
13. MCD : Modèle Conceptuel de Données [↑](#footnote-ref-13)
14. MLD : Modèle Logique de Données [↑](#footnote-ref-14)
15. UX : User Expérience [↑](#footnote-ref-15)
16. MVC : Modèle Vue Controller [↑](#footnote-ref-16)
17. CI/CD : Continuous Integration, Continuous Deployment [↑](#footnote-ref-17)
18. RGPD : Règlement Général sur la Protection des Données [↑](#footnote-ref-18)
19. Règlement général sur la protection des données (RGPD), article 32, 2023, <https://www.gdpr-expert.eu/article.html?id=32> [↑](#footnote-ref-19)
20. Documentation officiel PHP, fonction « password\_hash », 2023, <https://www.php.net/manual/fr/function.password-hash.php> [↑](#footnote-ref-20)
21. SSL : Secure Sockets Layer [↑](#footnote-ref-21)
22. FTP : File Transfer Protocol [↑](#footnote-ref-22)
23. ETL : Extract, Transform, Load [↑](#footnote-ref-23)