UNIVERSITÉ DU QUÉBEC
À CHICOUTIMI

Rapport de stage

Participation au développement et à l'optimisation de l'interface utilisateur des applications web

Lucas Senis

1537 – Maîtrise en informatique (jeux vidéo)

Je souhaiterais remercier Damien Brun *pour m'avoir encadré et guidé dans ma recherche de stage et plus généralement dans l'ensemble de mes démarches,*

mon tuteur Alexandre Jumeline *pour m'avoir fait découvrir l'entreprise et ses partenaires ainsi que pour m'avoir inclus dans l'équipe en qualité de chef de projet,*

Aurélien Wulsztat *et Wendy* Lautier *pour m'avoir accueilli au sein d'Akeros et s'être assurés de mon intégration tant technique que sociale,*

Christine Carlier *pour m'avoir accompagné dans mes démarches administratives du début à la fin de mon stage,*

Clément Lalu *et Philippe* Bousselier*, collègues développeurs avec qui j'ai interagi au jour le jour et qui ont soutenu ma progression technique,*

Chih-Chi Lin *avec qui j'ai pu partager l'expérience d'être stagiaire à Akeros,*

et enfin ma famille proche, qui a su me soutenir dans les bons moments comme dans ceux plus difficiles (en particulier durant ma recherche de stage) et m'aider à aller de l'avant en toutes circonstances.

Table des matières

Introduction

Ce stage s'inscrit dans la continuité des deux précédents : développeur *full-stack* en 2022, *back-end* en 2023, et maintenant orienté *front-end* (avec certains aspects *back-end*). Cela m'a permis d'affiner mes compétences en développement web, dans l'optique de travailler dans ce domaine à l'issue de mes études.

J'ai commencé à chercher un stage début Février 2024, en déposant des candidatures spontanées dans des entreprises québécoises, portfolio de mes projets à l'appui. N'ayant toujours pas trouvé de stage à l'issue de mes cours théoriques fin Juin, je suis retourné en France pour y poursuivre mes recherches, que j'ai d'ailleurs élargies géographiquement en incluant la France en plus du Québec. J'ai commencé à être plus présent sur les réseaux de recherche d'emploi : actualisation de mon profil et activation d'alertes sur Linkedin, Indeed et JobTeaser ainsi que message annonçant ma recherche de stage sur Linkedin. En plus de ces démarches, j'ai effectué une refonte de mon portfolio (jusque là fait à la main) avec WordPress pour lui donner un aspect plus professionnel, que j'ai rempli de manière conséquente : ajout de mon cursus et des technologies que je connais (qui n'étaient auparavant accessible qu'en ouvrant mon CV). Ma recherche de stage durant de plus en plus longtemps, j'ai initié quelques projets personnels afin de continuer à programmer et développer mes compétences en développement web, ce qui m'a permis d'étoffer mon portfolio. Au mois d'Octobre, j'ai participé au forum Ingénib organisé à l'ENSEIRB-Matméca (mon école d'attache en France), durant lequel j'ai rencontré plusieurs entreprises. N'ayant pas trouvé de stage début 2025 par des recherches classiques, je me suis tourné vers mon réseau et celui de mes proches.

J'ai découvert Akeros par l'intermédiaire d'un contact en commun avec son président. Ce dernier m'a alors accordé un entretien RH/technique simultané avec lui-même ainsi que la responsable administrative dans les locaux de l'entreprise. Au cours de cet entretien, il m'a présenté les projets de l'entreprise et j'ai dû résoudre un problème algorithmique dans le langage de mon choix pour montrer mon niveau. À l'issue de l'entretien, j'ai reçu une proposition de stage que j'ai acceptée. Le temps de remplir la convention et de la faire signer par toutes les parties, j'ai pu commencer mon stage le Lundi suivant mon entretien (le 3 Mars). Si mon stage

devait initialement se terminer le 23 Juillet à l'issue des 675h de travail exigées par l'UQAC. Une extension du stage a été décidée d'un commun accord entre Akeros, l'UQAC et moi-même (détaillée en sous-section I.3.3). Cette extension qui a repoussé la date de fin de mon stage au 5 Septembre m'a permis de poursuivre mon apprentissage au sein d'Akeros et de continuer à participer aux projets en cours.

Première partie

Compte-rendu

I.1 Sujet du stage

I.1.1 ISI®

L'application ISI® a pour but de systématiser des séquences d'opérations physiques ou numériques de natures différentes. Ce programme permet d'automatiser des chaînes de production en ordonnançant les instructions d'une machine à la suivante.

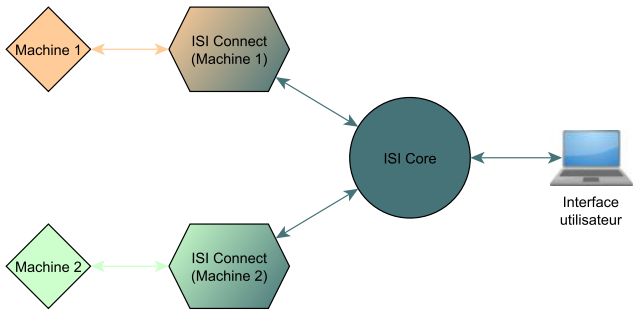

FIGURE 1 – Architecture logicielle simplifiée d'ISI®

La version 2 d'ISI® fonctionne selon une architecture modulaire organisée autour du module *ISI Core* (ainsi que plusieurs autres modules internes), assurant les tâches fondamentales comme l'ordonnancement des tâches de production, l'accès à la base de données et la gestion des droits. Les modules *ISI Connect* sont une couche d'abstraction et servent d'interface entre les systèmes (qui utilisent leur propre protocole de communication) d'une part et le reste du logiciel ISI® d'autre part. Un système peut être une machine ou un programme (même une autre instance d'ISI®). Pour chaque système géré par ISI®, un module *ISI Connect* est implémenté. Ceux-ci peuvent être utilisés pour produire des statistiques ou envoyer des notifications à l'utilisateur, et permettent également d'envoyer des ordres aux machines.

La version 3 d'ISI® marque une refonte conséquente de l'architecture logicielle. Les modules *ISI Connect* restent rétro-compatibles, mais l'ensemble des modules internes à ISI® ont été rassemblés avec *ISI Core* au sein d'un unique module *ISI Dashboard* contenant à la fois le *front-end* et le *back-end* générique à toute instance d'ISI®. Les aspects plus spécifiques sont gérés dans une surcouche propre à chaque utilisation d'ISI®. Par exemple, il y a un module dédié au fonctionnement d'ISI® pour les machines de LUCAS : ce module gère les systèmes *Eazy compact* et *Smart Driller*.

I.1.2 *Laser monitoring*

Ce projet constitue une application indépendante de l'architecture principale d'ISI®. Il vient se greffer directement à un laser de découpe de métaux de l'entreprise Mazak, comme le ferait un module *ISI Connect*, pour générer un historique ainsi que des statistiques relatives à ce laser. Sur la page principale (voir figure 2), on peut voir l'état du laser en temps réel (par exemple la consommation ou des indicateurs physiques comme la pression) et des informations sur le programme en cours (nom, estimation de l'heure de fin, etc.).

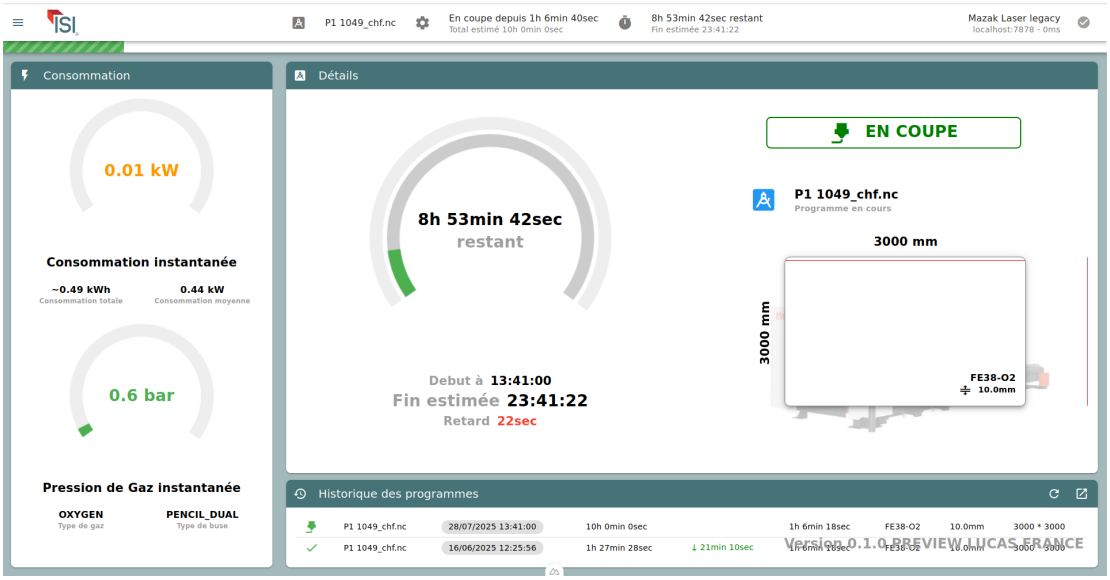

FIGURE 2 – Page d'accueil du logiciel *Laser monitoring*

I.2 Milieu

Akeros est une entreprise fondée en 2016, spécialisée dans la programmation d'applications web pour l'industrie. Nous y travaillons dans un *open space* partagé avec une entreprise de nettoyage de façade par drones nommée FlyRénov. Bien qu'Akeros et FlyRénov constituent des entités distinctes, les deux entreprises s'entendent très bien et il est fréquent que l'une demande un conseil ou un service informel à l'autre, ou même dans le cadre d'un partenariat officiel.

FIGURE 3 – Bâtiment hébergeant entre autres les entreprises Akeros et FlyRénov

Le principal partenaire d'Akeros est LUCAS Smart Tower Systems, une usine située à Bazas, en France. Lucas Smart Tower Systems est une branche de Lucas Robotic Systems, entreprise produisant majoritairement des axes linéaires.

Depuis 2022, Akeros et LUCAS travaillent ensemble à l'automatisation des chaînes de productions en poussant la modularité des machines à son maximum dans le cadre du projet *Smart Tower System*. LUCAS produit des tours de stockage de tôles qui viennent s'adapter à plusieurs machines : un laser de découpe produit par la société Mazak, et un *Driller* produit par LUCAS. Akeros programme la solution logicielle ISI® qui assure l'interopérabilité de ces systèmes. ISI®

est actuellement déployé chez des clients en France et aux États-Unis qui utilisent les tours produites par LUCAS. Le laser Mazak fait également l'objet d'un autre projet indépendant moins imposant qu'ISI®, *Laser monitoring*.

L'équipe d'Akeros est constitué de 6 personnes dont 3 développeurs à temps plein. Durant mon stage, nous étions deux stagiaires en plus des 6 employés fixes : je travaillais au sein de l'équipe de développeurs et l'autre stagiaire assistait la directrice générale dans les relations commerciales et la communication. Dans l'organigramme présenté en figure 4, tuteurs et stagiaires sont reliés par une flèche en pointillés. Les équipes de direction, de développement, commerciale et administrative sont respectivement représentées en rouge, vert, bleu et jaune.

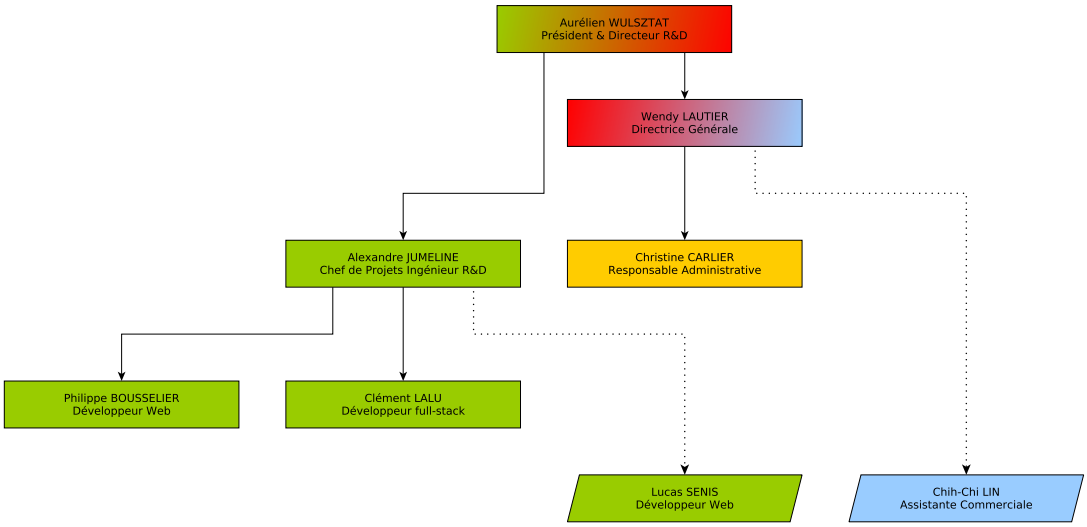

FIGURE 4 – Organigramme d'Akeros

Mon responsable au sein de l'entreprise est ingénieur chef de projets de l'équipe de développeurs. En plus de prendre part au projet en tant que développeur, il est responsable de la planification et de l'organisation des points d'avancement au fil du projet. Avec le président d'Akeros, ils forment également l'équipe R&D de l'entreprise et conçoivent les notions plus abstraites d'ISI comme la logique d'ordonnancement des opérations entre les machines.

I.3 Déroulement

I.3.1 *Laser monitoring*

Durant la première moitié de mon stage, j'ai travaillé sur le projet de *Laser monitoring* afin de me familiariser avec les technologies utilisées à Akeros, notamment le *framework* NuxtJS. J'ai commencé par implémenter les différentes requêtes et routes d'API nécessaires aux graphiques de la page de statistiques, qui étaient jusque là simulés avec des valeurs *mock*.

Parmi les graphiques à générer, certains nécessitaient des données qui n'étaient pas encore enregistrées par le logiciel : la consommation d'électricité ou de gaz, ainsi que les temps de pause. Si les données relatives à la consommation étaient bien affichées dans la vue "instantanée" présente sur la page d'accueil du logiciel (voir figure 2), les temps de pause n'étaient pas du tout implémentés et j'ai conçu en autonomie la manière de les stocker en base de données.

Enfin, nous avons souhaité déployer cette application sous la forme d'un exécutable plutôt qu'une page web accessible par navigateur. Pour cela, j'ai intégré le *framework* ElectronJS au projet et paramétré la compilation pour Linux et Windows à l'aide du module `electron-builder`.

I.3.2 Script d'installation d'ISI®

J'ai travaillé à plusieurs reprises sur un script d'installation d'ISI® ainsi que le tutoriel qui l'accompagne. J'ai d'abord été chargé de retravailler un script pré-existant. J'ai entre autres rajouté des retours dans le terminal pour que l'utilisateur puisse en suivre la progression, ainsi qu'une vérification de la connexion internet avant l'installation.

J'ai également rédigé un tutoriel pdf visuel pour accompagner l'utilisateur durant les différentes étapes de l'exécution du script. ISI® étant destiné à des clients français ou américains, ce tutoriel comporte une section dédiée au changement de langue du système d'exploitation.

Plus tard dans mon stage, alors que je travaillais sur la v3 d'ISI®, un problème d'installation a été rencontré chez un client sous KDE (un environnement de bureau alternatif à Gnome) après avoir exécuté le script, qui était uniquement prévu pour Gnome. J'ai donc repris le script (et le tutoriel) et ajouté l'installation de Gnome.

I.3.3 ISI®

Après avoir terminé une première version du projet *Laser monitoring*, j'ai rejoint les autres développeurs sur la programmation de la solution logicielle ISI®. Akeros mène actuellement deux projets principaux relatifs à ISI® : la transition vers ISI® v3 évoquée plus haut, ainsi que le développement d'un module *ISI Connect* pour monitorer le système *Smart driller*. J'ai effectué de nombreux tests sur les fonctionnalités impliquées et apporté des correctifs au différents bugs que j'ai pu rencontrer. Après m'être familiarisé avec le projet, j'ai également implémenté quelques nouvelles fonctionnalités.

Akeros, l'UQAC et moi-même avons convenu d'une prolongation de stage. Celui-ci devait initialement se terminer le 23 Juillet, au terme des 675h demandées par l'UQAC. Cependant, une démonstration du *Smart Driller* et de la version 3 d'ISI® aura lieu début Septembre et il nous a semblé profitable de prolonger le stage jusqu'à la durée maximum autorisée en France, soit 6 mois. Mon stage se termine donc le 5 Septembre. Durant cette extension de stage, j'ai corrigé divers bugs remontés par nos clients aux États-Unis ou en France. De plus, j'ai effectué plusieurs déplacements à l'usine de LUCAS à Bazas pour pouvoir développer au plus près des machines pilotées par ISI®. Je travaille également à la transition de Nats à Redis (voir sous-section II.1.2) pour l'échange de messages et d'événements entre les modules *ISI Connect* et *ISI Dashboard*. En parallèle de ce travail, j'étudie aussi la communication entre le *back-end* et le *front-end* en utilisant les *websockets* Nitro, une technologie intégrée à NuxtJS à titre expérimental et qui simplifie la création de *websockets* dans l'application. Le directeur R&D a fait appel à moi pour cette tâche car le projet de *Laser monitoring* sur lequel j'ai commencé mon stage fonctionnait également par *websockets*.

I.4 Objectifs et résultats

I.4.1 *Laser monitoring*

Les objectifs de ce stage étaient multiples : dans un premier temps, dans le cadre du projet *Laser monitoring*, il était question de transformer une preuve de concept pré-existante en un produit plus abouti.

Mon premier objectif était de remplacer les valeurs *mock* utilisées pour les statistiques par des valeurs réelles. La page de statistiques comporte actuellement deux onglets : un pour les programmes et un pour les alarmes. Nous envisageons de créer un troisième onglet pour séparer la consommation (d'électricité ou de gaz par exemple) des programmes lorsque nous aurons plus de statistiques rentrant dans cette catégorie. En plus de la sélection du type de statistiques en naviguant entre les onglets, il est possible de sélectionner la période sur laquelle on veut examiner celles-ci. Par défaut, cette période va de mois en mois, mais il est possible de la régler d'un jour à un autre.

- Statistiques relatives aux alarmes

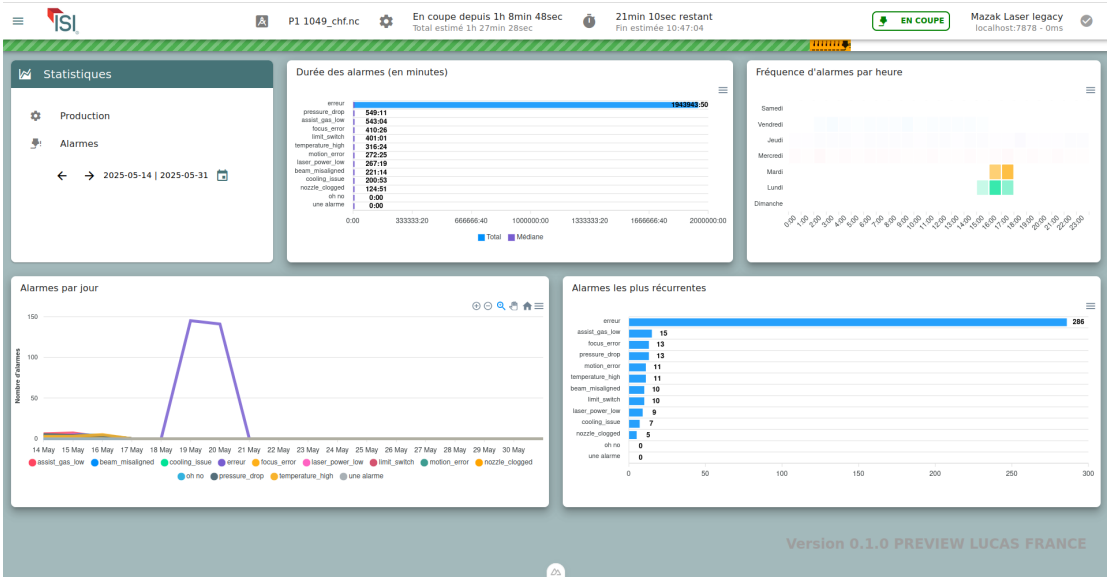

FIGURE 5 – Page des statistiques relatives aux alarmes

— durée des alarmes (durées totale et médiane par alarme sur la période donnée)

— fréquence d'alarmes par heure (carte thermique)

— alarmes par jour

— alarmes les plus récurrentes (classement des alarmes les plus fréquentes sur la période donnée)

- Statistiques relatives aux programmes

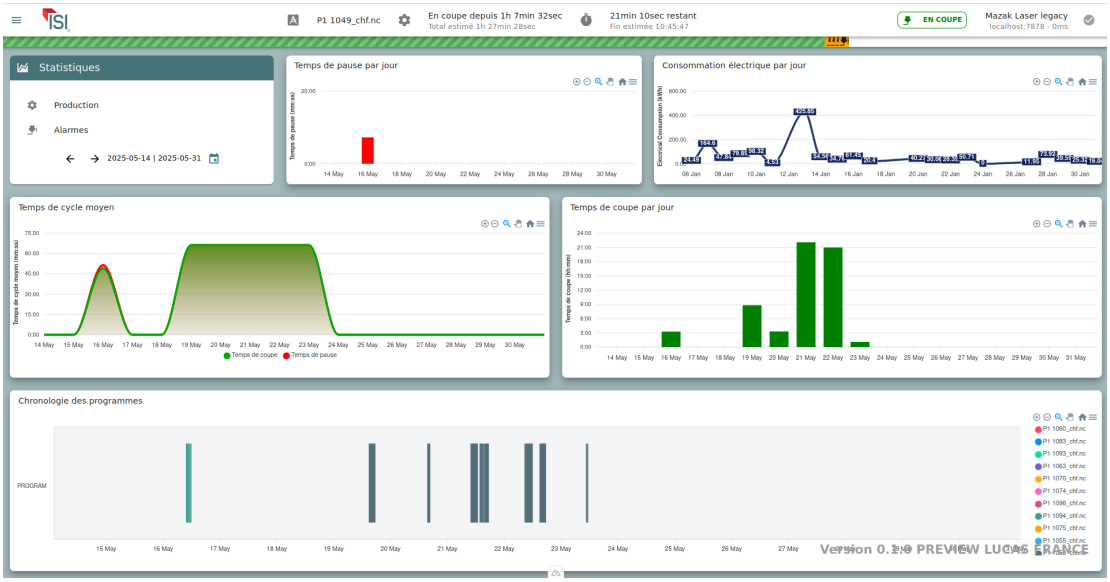

FIGURE 6 – Page des statistiques de production

— temps de cycle moyen

— chronologie des programmes (frise chronologique)

— temps de coupe / temps de pause

Sur le principe, ces deux graphiques sont similaires et affichent respectivement le temps de coupe et de pause par jour du laser. Le temps de coupe était assez simple à obtenir, puisque le laser envoie directement l'information du temps d'exécution d'un programme. Les temps de pause ont fait l'objet d'une réflexion plus poussée, développée en sous-sous-section II.1.1.1.

— consommation électrique

Il s'agit des dernières statistiques que j'ai implémentées. Pour cela, j'ai dû ajouter un système d'enregistrement de la consommation en base de données (initialement, seule la consommation instantanée était affichée sur la page d'accueil). La fréquence d'enregistrement de la consommation est paramétrable *via* une variable d'environnement.

En plus de mon travail sur les statistiques, j'ai également ajouté une nouvelle page à l'application *Laser monitoring*, qui est dédiée aux pièces découpées. En effet, un des principaux axes d'évolution actuels d'ISI® est le suivi des pièces de la matière première à la pièce finale en passant par les différentes machines de production. Bien qu'indépendant d'ISI® d'un point de vue de la programmation, *Laser monitoring* s'inscrit également dans cette philosophie. La base de données, les routes d'API et le *front-end* relatifs aux pièces sont fonctionnels, mais nous ne disposions pas des données nécessaires pour implémenter l'analyse des trames envoyées par le laser. À ce jour, cette fonctionnalité est donc encore expérimentale et n'est visible qu'en utilisant des valeurs *mock*.

Vers la fin de mon travail sur ce projet, il m'a été demandé de transformer l'application web en un exécutable indépendant. Pour cela, j'ai utilisé le *framework* ElectronJS (développé en sous-sous-section II.1.1.2) et produit des exécutables pour Windows et Linux. Nous souhaitions également rendre l'application compatible avec MacOS. Or, il est nécessaire de compiler sous MacOS pour produire un exécutable pour ce système d'exploitation, et nous ne disposions pas d'un ordinateur adapté.

I.4.2 ISI®

Mon travail sur ISI® a surtout consisté à tester les fonctionnalités déjà présentes de la v3 et corriger les éventuels problèmes trouvés à des fins de fiabilisation du code, en vue de déployer ISI® v3 vers la fin de mon stage. Suite à des demandes client supplémentaires, le cahier des charges a cependant évolué et ISI® v3 ne sera finalement déployée qu'après la fin de mon stage. Dans le cadre du développement du driller, j'ai aussi été amené à faire des correctifs sur ISI® v2.

Les modifications que j'ai effectuées étaient principalement dans le projet propre aux machines de LUCAS, mais j'ai également pris part aux modules génériques d'ISI®, *ISI Core* (pour la v2) et *ISI Dashboard*.

J'ai également implémenté de nouvelles fonctionnalités, parmi lesquelles des indicateurs visuels lorsqu'un outil est cassé et des champs de recherche dans la liste des outils. Enfin, j'ai grâce à ma maîtrise des langues complété les traductions du module `i18n` intégré de NuxtJS.

Comme évoqué plus haut, Akeros assure un travail de maintenance d'ISI® v2 en parallèle du développement de la v3 et du driller. Aussi, j'ai été chargé de consigner l'ensemble des modifications apportées à ISI® v2 depuis qu'Akeros travaille sur la v3 (fin 2024) afin de pouvoir reporter dans ISI® v3 les modifications pertinentes. Ainsi, les changements apportés à *ISI Core* et *ISI Dashboard* v2 sont à reporter dans *ISI Dashboard* v3. La plus grande partie de ces modifications concerne cependant la surcouche spécifique aux machines de LUCAS (j'ai également consigné ces modifications en vue les reporter). J'ai terminé le report des modifications que j'avais consignées initialement, désormais il reste seulement à effectuer les modifications à la fois en v2 et en v3 au fil de l'eau.

Deuxième partie

Réflexions

II.1 Analyse

II.1.1 *Laser monitoring*

II.1.1.1 Temps de pause

L'enregistrement des temps de pause était un aspect nécessaire du projet *Laser monitoring* puisque le graphique des temps de pause faisait partie des fonctionnalités requises établies en début de projet. Or, cet enregistrement ne se faisait pas encore au moment où j'ai repris le projet (il n'est d'ailleurs pas présent dans ISI® à ce jour). On m'a accordé une grande liberté en ce qui concerne son implémentation, j'ai donc conçu une table dans la base de données pour représenter l'état du laser au fil du temps en suivant un modèle de liste chaînée. La table `phases` comporte 4 champs (sans compter l'identifiant de chaque entrée) :

- `startDate` : un *timestamp* en secondes correspondant au début de la phase
- `next` : auto-référence vers l'identifiant d'une autre entrée de la table des phases, qui correspond à la phase suivante
- `program_id` : l'identifiant du programme en cours (référence la table `programs`)
- `status` : prend une valeur entière entre 0 et 3 inclus, correspondant aux valeurs de l'`enum` `LaserState` :
  - `Cutting` : programme en cours de coupe
  - `Paused` : programme actuel mis en pause
  - `Ready` : aucun programme en cours
  - `Unknown`

Lorsqu'on ajoute une phase, elle prend les valeurs suivantes :

- `startDate` : le *timestamp* actuel

— `next` : `NULL`

— `program_id` : l'identifiant du programme en cours, `NULL` si aucun programme n'est en cours

— `status` : le statut actuel du laser

De plus, si une phase est en cours lors de l'ajout d'une nouvelle phase, son champ `next` prend pour valeur l'`id` attribué à l'entrée que l'on vient de créer.

Avec cette représentation des phases, on peut effectuer les opérations basiques suivantes :

— obtenir la phase en cours (s'il y en a une) : il s'agit de la seule dont le champ `next` vaut `NULL`

— obtenir la durée d'une phase : comparer le champ `startDate` d'une entrée avec le champ `startDate` de l'entrée référencée par le champ `next` (si `next` vaut `NULL`, on remplace cette valeur par le timestamp actuel)

— symboliser une erreur sur une phase : une phase est dite corrompue si elle se référence elle-même par son champ `next`. Ce cas de figure théoriquement impossible permet de signifier que cette phase est en erreur, tout en l'ignorant dans les calculs de temps de pause (d'après la définition ci-dessus, sa durée vaut 0).

Ainsi, on peut obtenir le temps de pause d'un programme en sommant la durée de toutes les phases le référençant avec `program_id` et ayant pour statut `Paused`.

Afin d'assurer la cohérence de cette table, et plus particulièrement de la liste chaînée basée sur les champs `id` et `next`, des vérifications sont effectuées au lancement de *Laser monitoring* :

— Si plusieurs phases sont en cours (leur champ `next` vaut `NULL`), toutes sauf la plus récente sont marquées comme corrompues. En utilisation normale, ce cas de figure ne se présente jamais mais cela constitue une sécurité supplémentaire au cas où la base de données aurait été modifiée en dehors du logiciel *Laser monitoring*. Après cette vérification, il ne peut y avoir que 0 ou 1 phase en cours.

— Si une phase est en cours, on ajoute une phase de statut `Unknown` entre cette phase et l'instant présent. Cela revient à ajouter deux entrées dans la table (et mettre à jour le champ `next` de la dernière entrée présente au redémarrage), comme indiqué en figure 7. Si aucune phase n'est en cours, seule la deuxième ligne est nécessaire.

id	startDate	next	program_id	status
...	...	x_1	...	...
x_1	*date d'arrêt du monitoring*	x_2	NULL	Unknown
x_2	*date actuelle*	NULL	*programme actuel*	*statut actuel*

FIGURE 7 – Entrées ajoutées à la table des phases au redémarrage de *Laser monitoring*

II.1.1.2 Déploiement du projet

Le projet *Laser monitoring* se veut beaucoup plus léger qu'ISI®, et nous souhaitions que cela se ressente aussi dans son utilisation. Aussi, j'ai étudié les possibilités qu'offrait le module ElectronJS pour transformer cette application web en un exécutable indépendant.

J'ai pour cela utilisé le module `nuxt-electron`, qui m'a permis d'obtenir une version fonctionnelle en mode développement assez rapidement. L'un des principaux défis dans la mise en place d'ElectronJS résidait dans la compilation de l'application en un fichier exécutable. On trouve trois principaux compilateurs d'ElectronJS : `electron-forge`, `electron-builder` et `electron-packager`. Dans un premier temps, j'ai souhaité utiliser `electron-forge`, qui est la solution de compilation officielle fournie par l'équipe de développement d'ElectronJS. Cependant, ce module est relativement peu répandu en comparaison avec `electron-builder` et j'ai finalement basculé sur ce dernier qui est bien mieux référencé. Grâce à `electron-builder`, j'ai pu compiler *Laser monitoring* pour Linux dans un premier temps, au format `AppImage`. Ensuite, j'ai également produit une version `.exe` pour Windows, que j'ai testée sur une machine virtuelle que j'ai mise en place sur mon ordinateur. Nous avions souhaité compiler l'application pour MacOS, mais cela requiert une machine tournant sous MacOS pour la compilation, ce dont nous ne disposons pas pour le moment.

Afin de remonter les potentielles erreurs survenant en production, j'ai également intégré Sentry au projet, qui capture les erreurs du *front-end* comme du *back-end* et les remonte à un serveur d'Akeros qui rassemble les projets utilisant ce module. Les erreurs sont remontées avec un certain nombre de tags personnalisés comme le type ou l'adresse du laser, ainsi que la pile d'appel.

Une fois ce système de suivi des bugs mis en place, nous avons envoyé l'application en bêta-test chez LUCAS. Grâce à Sentry, 4 bugs mineurs nous ont été remontés et leur correction en a été simplifiée.

II.1.2 ISI®

Pour fonctionner, ISI® nécessite plusieurs services :

— `mysql` : la base de données principale d'ISI®

— `redis` : le SGBD qui gère les données plus "éphémères" comme les missions ou les queues d'exécution de programmes sur les différentes machines

— `nats` : l'agent de messages par lequel les évènements sont communiqués entre les modules *ISI Connect* et *ISI Dashboard*.

— `ftp`/`sftp` : ISI® permet à l'utilisateur de téléverser des fichiers (par exemple la documentation des machines ou du logiciel)

Le SGBD Redis a été privilégié plutôt que MySQL pour les données éphémères du fait de sa rapidité d'exécution. En effet, Redis effectue des opérations en mémoire vive tandis que MySQL écrit et lit directement sur le disque.

Pour pouvoir travailler sur ISI®, j'ai déployé l'ensemble de ces services sur un conteneur Docker, que j'ai mis en place avec la syntaxe `docker-compose`, qui permet de générer un conteneur Docker (télécharger et installer les images, monter les volumes et les réseaux) à partir d'un fichier YAML. J'ai ensuite pu lancer *ISI Dashboard* et les différents modules *ISI Connect* en parallèle (avec leurs simulateurs respectifs) et commencer à tester l'application.

Dans le cas d'ISI® v3, un seul script permet de lancer à la fois le *front-end* et le *back-end*. En ce qui concerne ISI® v2, ceux-ci doivent être lancés séparément.

II.2 Impressions personnelles

II.2.1 Situations rencontrées

Durant mon premier mois de stage, j'ai accompagné les autres développeurs lors d'une rencontre avec notre partenaire LUCAS à Bazas. Mon tuteur m'a alors fait visiter l'usine et j'ai pu réaliser à quel point les machines contrôlées par notre logiciel étaient imposantes. À titre de comparaison, le pavillon de l'écomobilité de l'UQAC est aussi grand que deux tours de LUCAS Smart Tower System, et haut comme une seule machine.

Lors des entretiens annuels de l'équipe au mois de Mai, j'ai moi aussi eu une entrevue avec le président et la directrice générale d'Akeros afin de faire un point sur ma première moitié de stage. Ils semblaient satisfaits de mes résultats techniques, mais on m'a reproché ma timidité. C'est donc un point que je me suis efforcé de travailler par la suite et je suis parvenu à bien m'intégrer à l'équipe, et même forger des amitiés, que ce soit lors de pauses café ou en mangeant tous ensemble le midi.

Afin de promouvoir le *Driller*, LUCAS participe à un *showroom* organisé par la société Mazak Optonics Corporation à Chicago. Afin de procéder à l'installation de la machine (et faire le tour des clients à proximité), le président d'Akeros a fait le déplacement. Or, un grand nombre de paramètres diffèrent de la production chez LUCAS, notamment la disposition des machines ou le système d'unités utilisé. Il est donc arrivé qu'il rencontre des problèmes inconnus jusqu'alors, que nous l'avons aidé à résoudre à distance avec l'équipe de développeurs.

Au fil du stage, j'ai pris part aux réunions commerciales régulières organisées par la directrice générale d'Akeros. Cela m'a permis d'avoir une meilleure vision d'ensemble de l'entreprise et de ses perspectives d'évolution. À l'occasion de ces réunions, nous discutions entre autres de ses déplacements dans d'autres entreprises afin d'étudier leurs besoins et de prendre contact avec de potentiels clients.

II.2.2 Apport du stage

II.2.2.1 Apports techniques

Grâce à ce stage, j'ai affiné mes compétences en développement web, et plus particulièrement le *framework* Vue.js. J'ai notamment pu apprivoiser le *meta-framework* NuxtJS, qui simplifie la communication entre un *back-end* Node.js et un *front-end* Vue.js.

À l'occasion de mon travail sur le script d'installation d'ISI®, j'ai mis en place une machine virtuelle, compétence que j'aurai souvent l'occasion de réinvestir à l'avenir. En effet, quelques semaines plus tard, j'ai eu besoin d'une autre machine virtuelle dans le cadre des tests de compilation avec ElectronJS (notamment pour Windows).

Par ailleurs, j'ai eu durant ce stage une utilisation quotidienne de GitLab, ce qui m'a permis de maîtriser `git` plus en profondeur que lors des projets effectués en cours. En effet, j'ai fait un usage intensif des *merge requests* pour la validation de code, ce qui était moins nécessaire sur les projets courts que j'ai menés dans le cadre de mes études. L'utilisation de git dans le cadre des cours se limitait principalement aux commandes `add`, `commit`, `pull` et `push` (et dans une moindre mesure `branch` et `checkout`). Durant ce stage, j'ai eu recours à `cherry-pick` pour reporter des correctifs d'une branche dans une autre tout en facilitant les `merge` futurs.

II.2.2.2 Carrière

Au cours de ce stage, j'ai pu avoir un meilleur aperçu du travail en équipe en entreprise que lors des précédents. En effet, j'avais jusqu'ici été assez autonome dans mes projets. Durant la première partie de ce stage, j'ai retrouvé cette même autonomie le temps de me familiariser avec les technologies utilisées pendant mon travail sur le projet *Laser monitoring*. En rejoignant l'équipe de programmation d'ISI®, j'ai pu me sentir davantage intégré à l'équipe, avec la possibilité d'échanger avec les autres développeurs lorsque l'un d'entre nous a une question ou un problème.

II.2.3 Préparation du stage

II.2.3.1 Formation universitaire

L'essentiel de ma formation à l'UQAC concernait la programmation de jeux vidéo. Si les technologies que j'ai utilisées sont bien différentes (en cours, je programmais majoritairement en C++ contrairement à JavaScript/TypeScript pendant mon stage), j'ai tout de même retrouvé certains thèmes de la programmation de jeux vidéo comme les aspects UX/UI.

D'un point de vue plus technique, j'ai pu mettre à profit ma formation à l'ENSEIRB-Matméca, et notamment le cours de programmation fonctionnelle durant lequel j'ai adopté de bonnes pratiques de programmation pour les langages de la famille ECMAScript.

II.2.3.2 Auto-formation

Ma recherche de stage s'étant prolongée après la fin de mes cours, j'ai entrepris de me former à de nouvelles technologies en parallèle de mes démarches. Recherchant un stage dans le développement web, je me suis particulièrement penché sur ces technologies, notamment les *frameworks* Vue.js et Angular, ainsi que Java pour le *back-end*. Vue.js est le *framework* que j'ai le plus investi, ayant également mené un projet personnel avec pendant l'été 2024.

Conclusion

Je garde de ce stage au sein d'Akeros un ressenti très positif. Ce stage de fin d'études m'a permis de mettre en pratique ce que j'ai pu apprendre dans le cadre de mes cursus à l'UQAC et à l'ENSEIRB-Matméca, ainsi que durant mes précédents stages ou dans le cadre de formations en autonomie.

J'ai également pu apprivoiser de nouvelles technologies comme NuxtJS, les machines virtuelles, et les conteneurs Docker.

Durant ce stage, j'ai pu avoir une vision plus complète du monde de l'entreprise que lors de mes précédents stages. En effet, j'ai pu réaliser les différentes implications du travail en équipe, que ce soit dans l'organisation du travail et la répartition des tâches ou bien la gestion de projet et la collaboration sur `git`.

Les multiples réunions commerciales organisées tout au long de mon stage et les interventions à l'usine de LUCAS m'ont aussi permis de mieux comprendre les principes de relation client et les impératifs que cela peut représenter d'un point de vue du développement.