

# Rapport de stage

Portage informatique de la solution Artiscan

**BECARNE Anthony**

Du 05 juillet 2021 au 31 décembre 2021

**Tuteur de stage :** Vincent BERTE

**Tuteur référent :** Paul COMPERE

**Établissement d'enseignement :** EPITECH Lille

**Entreprise d'accueil :** AQUILAB Loos

## REMERCIEMENTS

Avant tout propos concernant ce rapport, je souhaiterais adresser mes remerciements à toutes les personnes qui m'ont aidé et qui ont contribué au bon déroulement de ce stage.

Mes remerciements s'adressent en premier lieu à **M. Vincent BERTE**, Ingénieur Développement et maître de stage, qui m'a suivi et aidé dans la réalisation de mes missions au cours de ces six derniers mois.

Je remercie **M. David GIBON**, Directeur Général pour m'avoir donné l'opportunité de réaliser ce stage et de m'avoir accueilli au sein de son entreprise.

Je remercie également **M. Ali DARWICH** et **M. Erwan DOUAILE**, Ingénieurs en Recherche et Développement pour l'aide précieuse qu'ils m'ont apportée au cours de mes missions ainsi que leurs conseils pour la rédaction de ce rapport.

Je remercie **M. Vincent LEFORT**, prestataire pour AQUILAB, pour l'aide qu'il m'a fourni sur certaines missions.

Je tiens aussi à remercier **Mme. Karine RUCKEBUSCH**, Attachée de Direction pour son accueil chaleureux et sa bonne humeur, et qui m'a permis de trouver rapidement ce stage.

Aussi, je remercie toute l'équipe d'AQUILAB, pour leur soutien, leur accueil et qui m'a permis de faire mon stage dans une ambiance très conviviale.

Je saisis cette occasion pour adresser mes remerciements à l'équipe pédagogique de **l'École Epitech Lille** qui nous donne l'opportunité de réaliser des stages lors de notre cursus.

## INTRODUCTION

Dans le domaine de la santé, l'IA est au cœur de la médecine du futur : prédiction d'une maladie et/ou de son évolution, recommandation de traitement personnalisé, chirurgie assistée par ordinateur etc. Grâce au recoupement d'un nombre croissant de données (big data), il est aujourd'hui possible d'utiliser des algorithmes d'apprentissage automatique (machine learning) afin d'élaborer des modèles prédictifs.

C'est dans le cadre de mon entrée en deuxième que j'ai effectué ce stage du 05 juillet au 31 décembre 2021 (6 mois). L'objectif du stage était de migrer l'application Artiscan depuis Visual Studio 2010 vers Visual Studio 2019. Cet objectif était accompagné de l'évolution de librairies externes au besoin.

L'entreprise AQUILAB réalise la conception, la production et la commercialisation de dispositifs, logiciels et matériels, études et services dans le secteur de la santé humaine. De plus, AQUILAB, située à Loos (Nord) et créée en 2000 s'est fait connaître grâce à ses logiciels spécialisés dans le contrôle qualité et l'évaluation en oncologie, radiothérapie et imagerie médicale. Elle agit donc activement dans le milieu de la radiothérapie pour proposer des solutions qui sont utilisées partout en France et en Europe.

Pour une question de lisibilité et de logique, j'ai décidé de découper ce rapport en quatre parties distinctes.

Dans un premier temps, je présenterai l'entreprise AQUILAB. Dans un second, j'étudierai la solution Artiscan ainsi que les raisons qui ont poussé à mettre en place mon stage. Dans un troisième temps, j'annoncerai les travaux que j'ai effectués durant ces six derniers mois. Finalement, je terminerai par les apports de ce stage, puis par mes préconisations pour poursuivre mon travail.

# Table des matières

<b>REMERCIEMENTS.....</b>	<b>2</b>
<b>INTRODUCTION .....</b>	<b>3</b>
<b>I. PRÉSENTATION DE L'ENTREPRISE AQUILAB .....</b>	<b>5</b>
A. Cadre historique.....	6
B. Cadre juridique.....	6
C. Les missions et les métiers.....	7
D. Les produits et services.....	8
<b>II. L'ENVIRONNEMENT DU STAGE .....</b>	<b>9</b>
A. La solution Artiscan.....	10
1. Présentation.....	10
2. Code source .....	13
3. Librairies externes.....	14
B. La dette technique .....	15
1. Définition .....	15
2. Dans le contexte d'ARTISCAN .....	16
3. Les conséquences .....	17
<b>III. TRAVAUX EFFECTUÉS.....</b>	<b>18</b>
A. Identification des besoins techniques .....	19
B. Développement .....	21
1. Mise à jour des librairies .....	21
2. Modification du code.....	23
C. Tâches périphériques.....	25
1. Mockup Artiscan .....	25
<b>IV. BILAN .....</b>	<b>27</b>
A. Les résultats .....	28
B. La suite de mon travail ? .....	28
<b>Conclusion.....</b>	<b>29</b>
<b>Bibliographie .....</b>	<b>30</b>
<b>Annexe : .....</b>	<b>31</b>

# I. PRÉSENTATION DE L'ENTREPRISE AQUILAB

## A. Cadre historique

AQUILAB est une PME française créée en 2000 par M. David Gibon et M. Philippe Bourel dans le but de donner suite à un besoin de sécurité et qualité du traitement de l'imagerie médicale et de la radiothérapie.

C'est après quatre années de recherche-développement que David Gibon et Philippe Bourel ont mis au point leur solution. Les logiciels d'AQUILAB permettent de garantir la constance des mesures des équipements, mais aussi d'assurer pleinement la cohérence des machines entre elles alors que, souvent, un centre de radiothérapie peut compter jusqu'à quatre ou cinq constructeurs différents.

Avec ses 20 années d'expérience, AQUILAB se positionne aujourd'hui en tant que leader sur le marché français et européen des logiciels de contrôle qualité des dispositifs médicaux de radiothérapie et d'imagerie médicale et compte désormais 350 clients en Europe.

## B. Cadre juridique

AQUILAB est une société par actions simplifiées (SAS) d'une vingtaine d'employés et présidée par David Gibon. Son siège se situe au Parc Eurasanté – Biocentre Fleming, 250 rue Salvador Allende à Loos (59120).

En raison de son statut et de ses services délivrés partout en Europe, AQUILAB se doit de respecter plusieurs normes :

- Le **marquage CE** : indiquant que les logiciels d'AQUILAB sont conformes aux exigences de la directive européenne 93/42/CE
- La certification **ISO 13485** : qui établit les exigences à un système de management de la qualité propre au secteur des dispositifs médicaux (DM)
- Les **Bonnes pratiques cliniques** (BPC) ainsi que les **directives de l'ICH** (International Council for Harmonisation)
- La loi **RGPD** (Règlement Général sur la Protection des Données) : qui vise à renforcer l'encadrement de la circulation et le traitement de données à caractère personnel des personnes physiques

## C. Les missions et les métiers

La mission d'AQUILAB est de proposer à ses clients des solutions de contrôle qualité qui améliorent à terme le traitement en radiothérapie tout en restant innovant sur le marché français et européen grâce à ses travaux de recherches.

En ce sens, les valeurs clé d'AQUILAB sont :

### La qualité

Afin de garantir à ses clients des dispositifs sûrs, innovateurs et efficaces, AQUILAB accorde une importance majeure à la mise en œuvre de la Politique du Contrôle Qualité.

### L'innovation

L'innovation est au cœur de l'entreprise. Une équipe se charge de la participation à des programmes de recherches. Ce qui permet une veille technologique et une conception de produits innovants.

### L'assistance

AQUILAB garantit un service client fiable et efficace pour chacun de ses services. Dès l'installation de ses équipements, une équipe se déplace sur site pour former les utilisateurs. C'est aussi pour apporter des réponses rapides et précises qu'une équipe d'ingénieurs d'application est disponible.

Le personnel d'AQUILAB est réparti dans différents pôles :

- Le pôle **Développement Product & Services** dont l'équipe est divisée entre ARTISCAN et ARTIVIEW. Elle est principalement composée d'ingénieurs R&D. Les Essais Cliniques sont quant à eux gérés par une équipe d'ingénieurs web assurant le développement de la solution ONCO PLACE.
- Le pôle **Customer Support** est composé d'ingénieurs d'application. Ce sont eux qui sont chargés de l'installation, la formation des utilisateurs et le suivi client.
- Le pôle **Sales & Marketing** chargé des ventes des services. Accompagné par les chefs de projet d'ARTISCAN et ARTIVIEW.
- Le pôle **Finance, HR & Quality** chargé de tout l'aspect administratif, comptabilité, ressources humaines et de la gestion de la qualité et des certifications.

## D. Les produits et services

**ARTIVIEW**



Figure 1. Cartouche ARTIVIEW

**« Améliorer votre expertise en optimisant la gestion des images patients et des données médicales »**

**ARTIVIEW** est un logiciel qui permet de préparer (fusion multi modalités, contourage 3D et segmentation Tomographique par Émission de Positrons automatique) et d'évaluer les plans de traitement en radiothérapie.

**ONCO PLACE**



Figure 2. Cartouche ONCO PLACE

**ONCO PLACE** est une plateforme web dont la fonction principale est le recueil de données et d'essais cliniques en radiothérapie. Cela comprend la centralisation, l'anonymisation, la structuration, l'analyse et le transfert des données.

Cette plateforme sert également de support à des formations sur le contourage en radiothérapie.

**ARTISCAN**



Figure 3. Cartouche ARTISCAN

**« La solution de Contrôle Qualité pour l'imagerie médicale et la radiothérapie »**

**ARTISCAN** est un logiciel d'imagerie et d'assurance qualité. Il permet de réaliser et de centraliser les contrôles qualité des équipements d'imagerie médicale.



## II.L'ENVIRONNEMENT DU STAGE

## A. La solution Artiscan

### 1. Présentation

**ARTISCAN** est une solution permettant de réaliser et de centraliser les contrôles sur des machines de mesures d'imagerie médicale. Cette solution se base principalement sur la radiothérapie mais supporte aussi les machines IRM, Scanners, Radiographies numériques et mammographies.



Figure 4. Accélérateur linéaire de particules

Un exemple de machine d'imagerie médicale serait l'accélérateur linéaire de particules (servant à la radiothérapie). La machine envoie des rayons sur une zone ciblée du patient et reçoit les images de ces analyses.



Figure 5. Fantôme de test

Pour effectuer les tests de contrôles qualités, on utilise des fantômes (voir Figure 5.) afin de s'assurer du bon fonctionnement des machines. Les fantômes sont des objets créés pour simuler des examens médicaux, ils permettent d'évaluer la qualité des images de radiographie.

Ce sont sur des fantômes de ce type, que tous les contrôles qualité d'ARTISCAN sont réalisés.

Aujourd'hui, ARTISCAN est la référence en « Imaging & Machine QA » en France mais elle est également présente dans le monde (Europe, Japon, Maghreb).

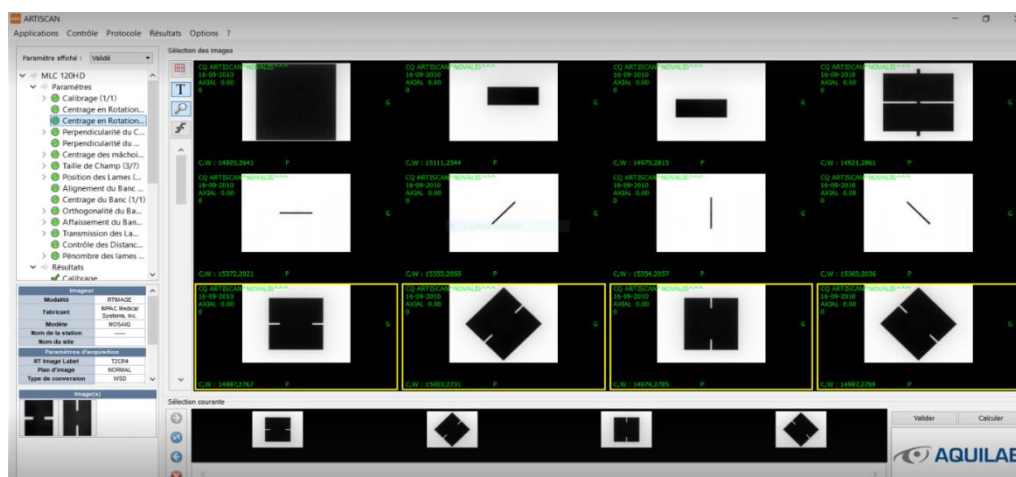


Figure 6. Capture d'écran du logiciel ARTISCAN

La réalisation de contrôles qualités sur le logiciel peut s'effectuer en quatre étapes :

## Étape 1 :

Après avoir lancé les mesures sur la machine d'imagerie, les images à tester sont récupérées.

L'utilisateur va entamer une procédure de création de contrôles depuis le logiciel ARTISCAN :

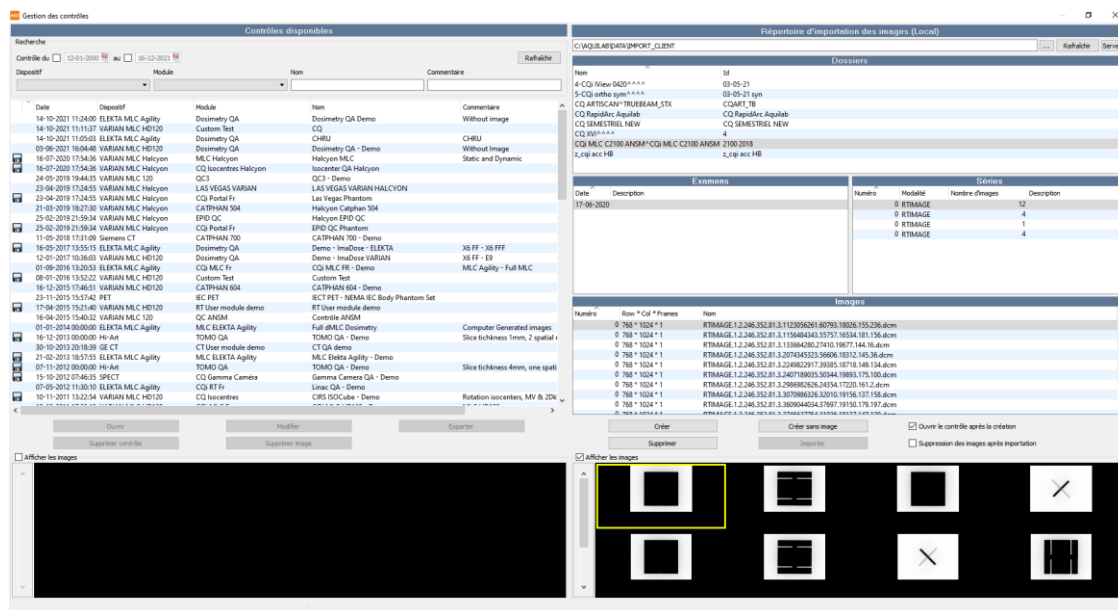


Figure 7. Création de Contrôle sur ARTISCAN.

## Étape 2 :

Une fois le contrôle créé, l'utilisateur va pouvoir sélectionner les images enregistrées par la machine et ARTISCAN procèdera à un ensemble de calcul sur chaque image (une image = un paramètre). L'utilisateur sera capable d'accéder à chaque résultat sur toutes les images (paramètres) prises en compte.

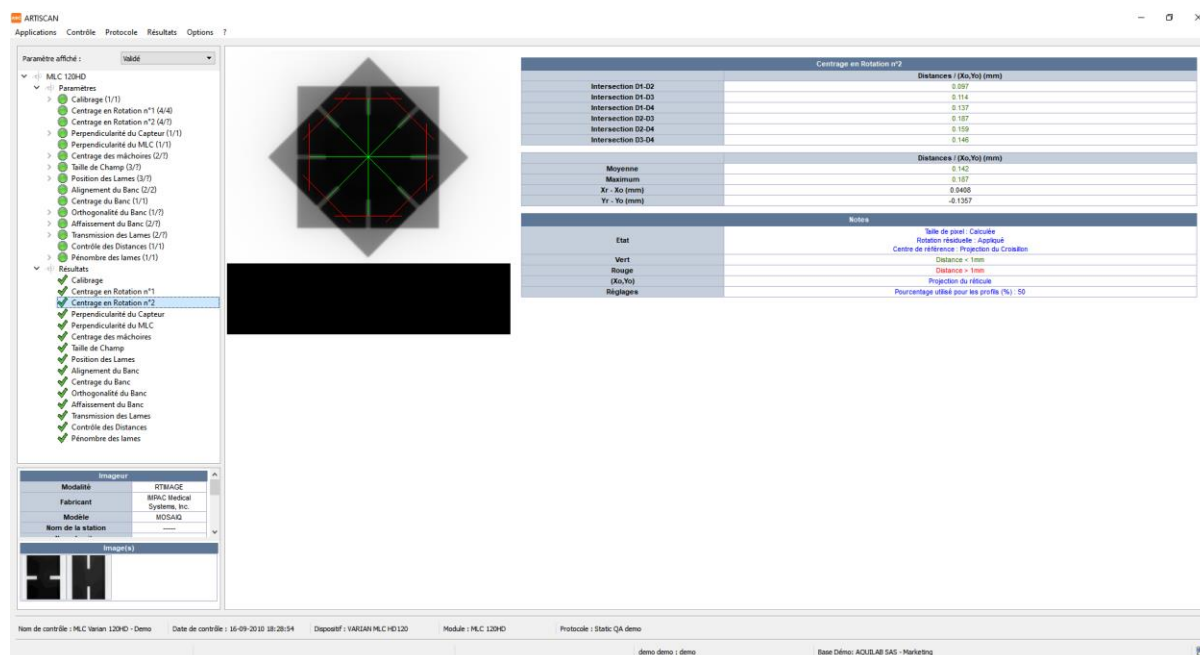


Figure 8. Résultat des calculs sur ARTISCAN

### Étape 3 :

Si l'utilisateur décide de sauvegarder cette base, ARTISCAN va envoyer tous les calculs vers une base de données via son serveur personnel. Cette base de données va pouvoir être interprétée et restituée à l'utilisateur via le WebReport d'ARTISCAN : un site internet regroupant toutes les bases de données de l'utilisateur.



Figure 9. Résultats des calculs sur le WebReport

### Étape 4 :

Le WebReport a stocké toutes les bases de données depuis la toute première sauvegarde de l'utilisateur. Celle-ci lui permettra de visualiser l'évolution des paramètres et de vérifier la cohérence de la machine par rapport à tous les calculs.

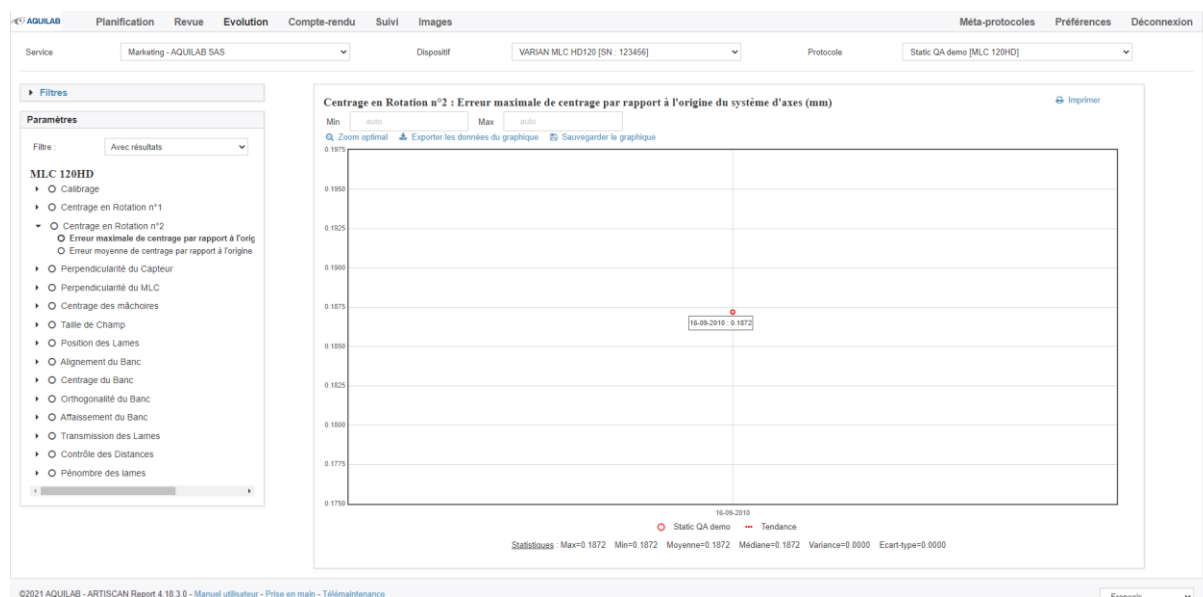


Figure 10. Évolution des bases de données en fonction du temps

## 2. Code source

ARTISCAN est un outil développé depuis une vingtaine d'années. Celui-ci est majoritairement programmé en C++, avec des normes et un paradigme de programmation d'époque. L'environnement utilisé est Visual Studio 2010 et donc le compilateur utilisé pour la solution est MSVC10<sup>1</sup>.

### **Pourquoi utiliser le langage C++ ?**

*Développé dans les années 1980, le langage C++ est dit de « bas niveau », c'est-à-dire qu'il est proche du langage machine. Le C++ fait partie des langages les plus puissants et les plus rapides, il correspond donc parfaitement à une utilisation pour un logiciel médical.*

Composé de 48 projets, ARTISCAN est basé sur une architecture Client-Serveur et peut être divisé en plusieurs sous-catégories :

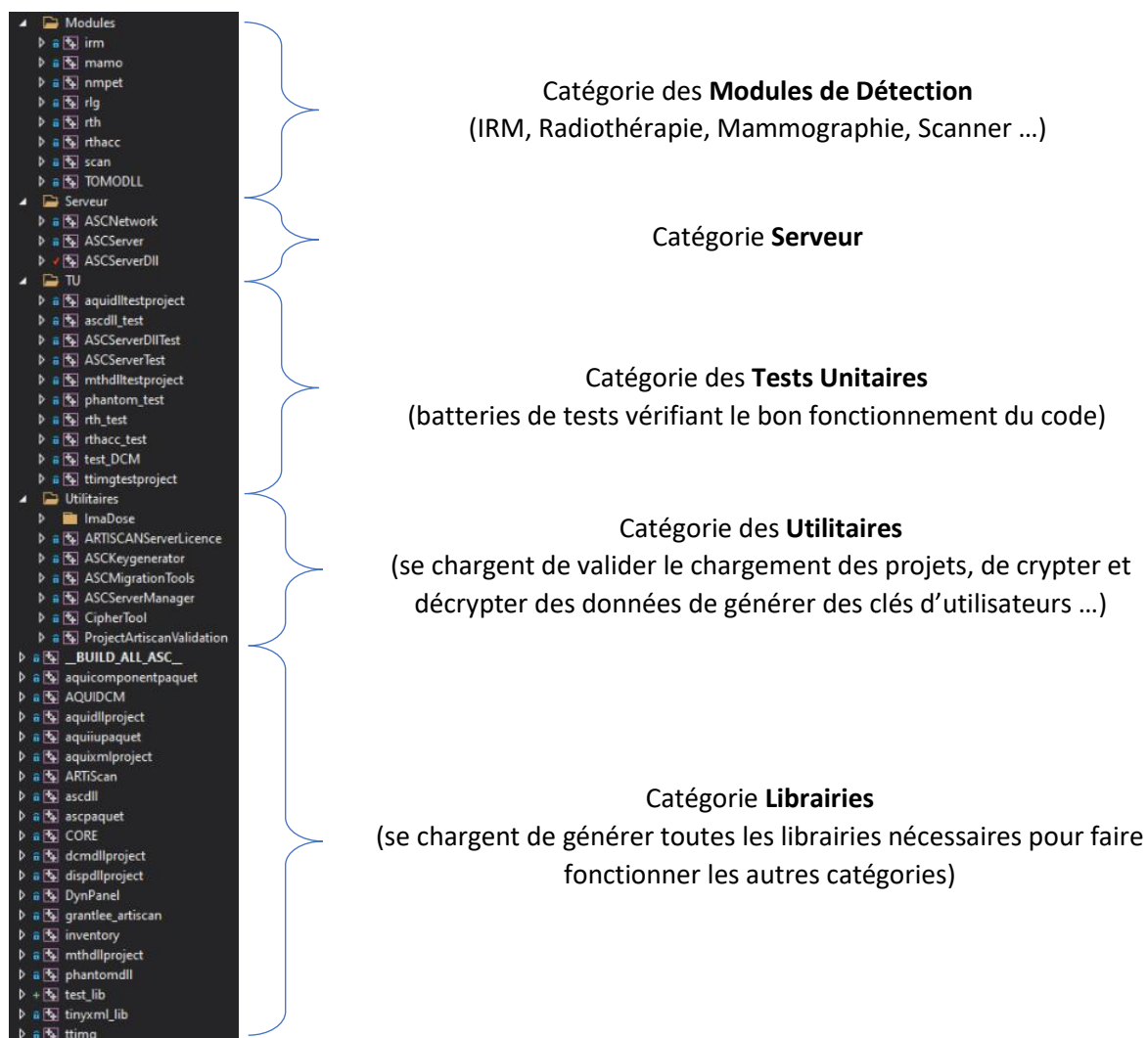


Figure 11. Liste des projets d'ARTISCAN

<sup>1</sup> Microsoft Visual C++ 10 (compilateur de C++)

Pour une question de clarté et pour visualiser l'architecture d'ARTISCAN, j'ai décidé de créer un organigramme regroupant l'ensemble des projets et leurs dépendances (voir Annexe 1).

### 3. Bibliothèques externes

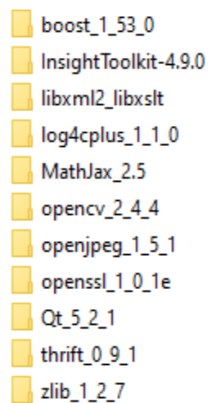
#### ***Qu'est-ce qu'une bibliothèque en informatique ?***

*Une bibliothèque est une collection de code et de fonctions prêtes à être utilisées par le développeur. En C++, la quasi-totalité des programmes utilisent la bibliothèque standard puisqu'elle est maintenant intégrée directement au C++. La bibliothèque standard du C++ ne prend pas en charge toutes les fonctionnalités, c'est pourquoi il est nécessaire d'utiliser d'autres bibliothèques selon les besoins.*

Le choix d'une bibliothèque peut être arbitraire, mais peut aussi être appuyé par plusieurs critères. Il est nécessaire de prendre en compte la fiabilité, l'efficacité mais aussi la documentation d'une bibliothèque avant de la choisir.

ARTISCAN est un logiciel médical. De ce fait, il est nécessaire pour l'utilisateur d'avoir un support visuel sur lequel travailler. La bibliothèque standard du langage C++ ne proposant pas de création d'interface visuelle, le seul moyen d'y parvenir est d'inclure une autre bibliothèque.

AQUILAB utilise une multitude de bibliothèques externes pour des tâches variées. On peut retrouver ci-dessous la liste des bibliothèques utilisées dans le développement de la solution ARTISCAN.



boost\_1\_53\_0  
InsightToolkit-4.9.0  
libxml2\_libxslt  
log4cplus\_1\_1\_0  
MathJax\_2.5  
opencv\_2\_4\_4  
openjpeg\_1\_5\_1  
openssl\_1\_0\_1e  
Qt\_5\_2\_1  
thrift\_0\_9\_1  
zlib\_1\_2\_7

Bien que chacune ait sa propre utilité, les bibliothèques peuvent cohabiter dans un même environnement. C'est le cas d'ARTISCAN ; un total de 11 bibliothèques différentes sont mises en place dans la solution.

Serveur, web, graphique ou compression de données, toutes les bibliothèques ont une place à part entière dans le logiciel.

Figure 12. Bibliothèques utilisées par ARTISCAN

Certaines utilisations contraignent les développeurs à utiliser une bibliothèque (comme pour une utilisation graphique), mais toutes celles présentées sur la Figure 12 ne sont pas indispensables.

## B. La dette technique

### 1. Définition

La dette technique est une métaphore inventée en 1992 par l'inventeur du concept de wiki, Ward Cunningham. Elle peut tout à fait être comparée au terme de dette en finance.

La dette technique a un impact non négligeable sur de nombreux points. Elle pose une limite à la capacité d'innovation et pèse sur les performances de l'entreprise. Néanmoins, les entreprises possèdent un coût d'opportunité si elles consacrent plus de temps, d'argent et d'autres moyens pour rectifier la dette technique plutôt que d'innover.

Les causes de cette dette technique sont nombreuses :

On retrouve parmi les causes les plus répandues une gestion bancal de la qualité du produit. Aucun contrôle ni tests ne sont mis en place pour garantir la fiabilité du logiciel, ce qui se traduit par une accumulation de dettes en continu. La deuxième cause provient d'une quelconque pression ; qu'elle soit due au temps, à la concurrence ou même aux moyens de l'entreprise, la rédaction d'un code propre et fiable est souvent laissée de côté.

D'autres causes comme la documentation insuffisante du code, le manque de qualifications ou bien la refactorisation du code trop tardive entraînent une dette technique considérable.

Beaucoup de solutions sont disponibles afin de minimiser une dette technique :

- En **limitant le nombre de technologies** utilisées. En effet, plus il y a de complexité dans une solution, plus la maintenance sera compliquée.
- En établissant des **normes de code strictes** : écrire du code maintenable, documenté et lisible.
- En faisant du **code review** au fil du développement. Le code review permet de faire un point sur le projet en s'assurant que les critères établis en amont ont bien été respectés.
- En mettant en place des **tests unitaires**, qui vont s'assurer que le code se comporte tout aussi bien d'une version à l'autre.
- En **s'adaptant aux besoins** et aux spécifications du projet. Il faut s'assurer que la technologie soit fiable et maintenue à jour régulièrement.

Malgré tous les efforts faits autour d'un projet, la dette technique reste inévitable.

## 2. Dans le contexte d'ARTISCAN

ARTISCAN, comme chaque projet en informatique, ne peut contourner la dette technique. Cette dette est la cause principale de la mise en place de mon stage. En effet, elle est présente dans cette application sous trois grands points.

### **Premier point : L'environnement de développement**

L'outil utilisé pour développer ARTISCAN est Visual Studio 2010 (plus de détails sur cet outil seront présentés dans la partie III.). Ce logiciel est sorti en avril 2010, il s'est fait remplacer par Visual Studio 2012, 2013, 2015, 2017, 2019 et 2022. En d'autres termes, ARTISCAN n'a pas changé d'environnement de développement depuis plus de 10 ans.

### **Deuxième point : Les librairies externes**

De la même manière que pour l'environnement, les librairies externes dans ARTISCAN n'ont pas évolué depuis un peu moins de 10 ans.

### **Troisième point : Le code source**

Tout comme l'environnement et les librairies externes, ARTISCAN n'a pas connu de grand changement d'architecture depuis ces 10 dernières années. À défaut de repartir sur des bases propres, solides et sans erreurs, c'est plutôt dans l'ajout de modules de détection qu'ARTISCAN s'est tourné. Bien que l'équipe de développeur continue sans cesse d'ajouter des correctifs depuis 10 ans, il n'empêche que certaines fragilités de l'architecture freinent l'ajout de nouvelles fonctionnalités.

Même si l'équipe d'ARTISCAN a mis en place des tests unitaires, a bien documenté le code et répond bien aux besoins du produit, il ne reste qu'elle accumule depuis des années, une dette technique importante.



### 3. Les conséquences

Bien qu'ARTISCAN possède une dette technique assez conséquente, il n'empêche que l'application fonctionne toujours et qu'elle continue à se vendre. Alors pourquoi devoir rembourser cette dette technique ?

#### **La performance**

En tant que logiciel de traitement d'images médicales, le facteur de la performance est au cœur des besoins. Aujourd'hui, ARTISCAN traîne des problèmes de rapidité de calculs et de communication entre le client et le serveur.

Pris indépendamment pour chaque machine de détection, ce manque de rapidité n'est pas grave. Mais dès qu'il s'agit d'un hôpital important, avec beaucoup de machines et de patients, la question des performances devient tout de suite cruciale.

#### **L'innovation**

De nos jours, la recherche médicale propose des outils toujours plus puissants. Le web à la pointe de la technologie, les nouveaux outils des librairies, les interfaces modernes ainsi que de nouveaux plugins de C++ voient le jour. L'intelligence artificielle prend une place importante dans la santé et ouvre de nouvelles portes dans l'imagerie médicale. Pour ARTISCAN, ces avancées technologiques sont une opportunité en or pour toucher un plus grand public.

Cependant, les espoirs d'intégrer ces outils dans l'application sont faibles tant que cette dette n'est pas prise en charge. En l'état actuel, il est presque impossible d'utiliser des fonctionnalités très puissantes. C'est pourquoi ARTISCAN doit rattraper son retard pour garder sa place dans le marché.

### III. TRAVAUX EFFECTUÉS

## A. Identification des besoins techniques

Le but premier de mon stage était de porter la solution ARTISCAN de Visual Studio 2010 vers Visual Studio 2019.

### ***Qu'est-ce que le portage d'une application en informatique ?***

- En informatique, porter une application signifie reprendre le code source du projet dans son environnement initial (ici Visual Studio 2010), puis de lui apporter les modifications nécessaires pour qu'il puisse fonctionner sur la plateforme de destination (ici Visual Studio 2019).

### **1. L'environnement de développement :**

#### ***Qu'est-ce que Visual Studio ?***

- Visual Studio (VS) est un EDI (de l'anglais IDE : Integrated Development Environment). Un EDI regroupe un ensemble d'outils spécifiques dédiés aux développeurs afin d'optimiser le temps de travail et la productivité en automatisant certaines tâches. Plus précisément, Microsoft Visual Studio est une suite de logiciels de développement pour Windows et MacOS conçue par Microsoft.

#### ***Quelle différence entre VS2010<sup>2</sup> et VS2019<sup>3</sup> ?***

- VS2010 a été publié le 12 avril 2010 tandis que VS2019 le 2 avril 2019. Ces 9 années de développement ont permis à VS de gagner en fonctionnalités des plus poussées avec de grandes avancées en termes de développement et d'outils. On retrouve parmi ces changements, la mise à jour du compilateur Microsoft Visual C++ (MSVC) propre à chaque version.

---

<sup>2</sup> Visual Studio 2010

<sup>3</sup> Visual Studio 2019

## 2. La norme ISO C++ :

Même si le langage est né à la fin des années 1970, il n'est normalisé que 20 ans plus tard, afin d'arrêter la profusion de versions C++ incompatibles. Le code C++ est standardisé par des normes internationales ISO<sup>4</sup>. Au fur et à mesure, elles sont devenues de plus en plus strictes et rigoureuses.

Bien que la liste des versions du standard C++ se trouve en bibliographie, voici une frise chronologique contenant les standard C++ ainsi qu'un exemple de l'évolution d'un membre.

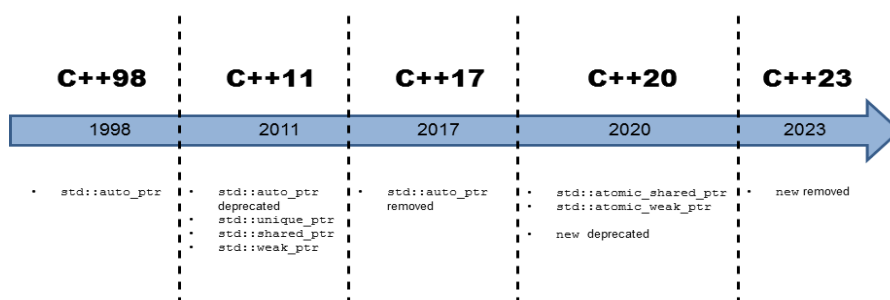


Figure 13. Frise chronologique des standards C++

## 3. La mise à jour des bibliothèques :

Les bibliothèques utilisées par ARTISCAN n'ont pas été mises à jour depuis un peu moins de 10 ans. Ce retard restreint l'accès à des nouvelles versions plus riches et plus fiables. La mise à jour de ces bibliothèques apporte des outils plus puissants et plus optimisés.

Le choix de la version des bibliothèques doit être pris en compte avant de la mettre à jour. Il ne faut pas prendre la dernière mise à jour parce qu'elle est la plus récente et possède le plus de fonctionnalités différentes. Il faut choisir une bibliothèque qui propose une version stabilisée et non bridée.

## 4. Le code source à adapter :

Le code source fait partie des points les plus compliqués à aborder. Il faut adapter le code pour qu'il respecte les règles de la norme C++14, puis le changer en fonction des bibliothèques qui vont être mises à jour. Même si le portage d'une application peut paraître simple, il y a de nombreux points techniques à explorer.

<sup>4</sup> International Organization for Standardisation

## B. Développement

### 1. Mise à jour des librairies

Comme expliqué précédemment, ARTISCAN utilise de nombreuses librairies afin d'effectuer des tâches complexes et variées.

J'ai donc étudié les cas possibles pour chaque librairie et j'ai ainsi pu mettre à jour celles qui le nécessitaient.

Parmi elles, on retrouve les suivantes :

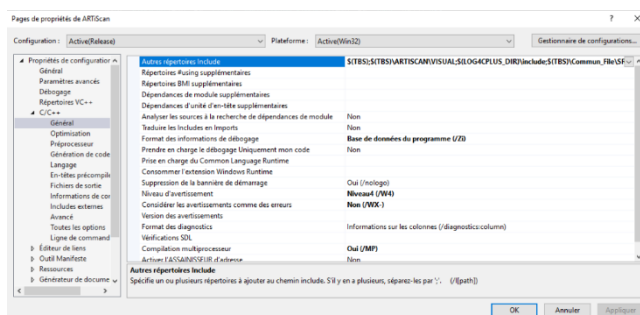
- Qt (version Qt 5.2.1 à Qt 5.15.1)
- Boost (version 1.53.0 à 1.70.0)
- Thrift (version 0.9.1 à 0.15.0)
- Libxml2 (version 2.6.0 à 2.7.8)
- Log4cplus (version 1.1.0 à 2.0.6)
- Zlib (version 1.2.5 à 1.2.11)

Certaines peuvent être mises à jour en téléchargeant simplement la librairie depuis son site officiel. D'autres nous obligent à recompiler les librairies en fonction des besoins du projet. C'est cette méthode que j'ai utilisée pour la majorité d'entre elles.

L'objectif de recompiler soi-même une librairie est de générer des fichiers « .dll » et/ou « .lib » en fonction de l'environnement de l'ordinateur (32bits ou 64bits).

En plus de ces fichiers « .dll » et « .lib », la librairie se compose de fichiers d'en-tête qui permet d'identifier des fonctions de la librairie que l'on va utiliser dans

Il reste donc à définir dans L'EDI<sup>5</sup> l'endroit où sont stockés ces deux types de fichiers pour que les fonctions utilisées soient reconnues.



Dans VS2019, il faut définir dans la catégorie « Autres répertoires Include » l'endroit où le dossier contenant les fichiers d'en-tête se trouve.

Figure 14. Définition des chemins vers les en-têtes

<sup>5</sup> Environnement de Développement (Visual Studio)

Puis dans les « Répertoires de bibliothèques supplémentaires », le dossier contenant les librairies.

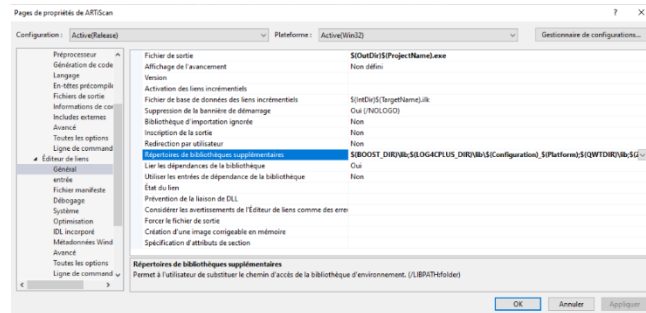


Figure 15. Définition des chemins vers les librairies

Les librairies sont compilables depuis un terminal de commande (ex : Boost), mais le sont aussi grâce à CMake (Thrift, Qt, Grantlee) ou téléchargeables directement GitHub (libxml2, libxslt).

CMake est un système de construction logicielle. Il est utilisé pour contrôler la compilation d'un projet à l'aide de fichiers de configurations qui ne dépendent ni de la plateforme, ni du compilateur.

GitHub quant à lui, est un service web d'hébergement et de gestion de développement de logiciels. Sur cette plateforme, on retrouve des dépôts de librairies par exemple.

## 2. Modification du code

ARTISCAN utilise des scripts python pour effectuer des actions très précises sur l'environnement de développement de VS.

### Les scripts python :

Ces scripts peuvent être utilisés pour :

- Lancer la solution ARTISCAN sous VS avec les variables d'environnement adéquates
- Exporter des fichiers de traductions
- Exporter des versions ARTISCAN
- Créer une archive de la solution

J'ai donc corrigé les scripts qui nécessitaient une mise à jour afin qu'ils fonctionnent avec VS2019. J'ai aussi créé un script qui exécute tous les autres en fonction des besoins de l'utilisateur.

```
full_setup.py > ...
26
27 all_files = [mandatories, update_files, build_files, launching] ## Ne spécifier que les tuples à exécuter
28
29 def execute_files():
30     for item in all_files:
31         for each in item:
32             file1 = "D:\ASC_VS2019\scripts\\" + each
33             os.system(file1)
34             time.sleep(1.2) # Temps d'attente entre chaque execution de script
35
36 def is_existing_file():
37     appropriate_file = 0 # Nombre de bons fichiers
38     nb_of_file = 0 # Nombre total de fichiers
39
40     for item in all_files:
41         for file in item:
42
43             ##### Mettre ici le chemin absolu vers le dossier scripts #####
44             absolute_path = "D:\ASC_VS2019\scripts\\" + file
45             #####
46
47             if not (os.path.isfile(absolute_path)): # Si le fichier n'existe pas
48                 raise ("The file {} is not in the current directory.".format(absolute_path)) # On raise une erreur
49             else:
50                 appropriate_file += 1 # Sinon on ajoute 1 au nb de fichiers qui existent
51                 nb_of_file += 1 # On va compter le nombre total de fichiers
52
53     if appropriate_file == nb_of_file:
54         execute_files()
55
56 if __name__ == '__main__':
57     is_existing_file()
58
```

Figure 16. Script "full\_setup.py"

### Le changement du code pour de meilleures performances

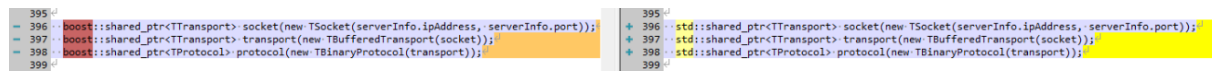
J'ai comparé le temps d'exécution entre les boucles de la librairie Boost (« BOOST\_FOREACH ») et les boucles « for ». Pour parcourir un tableau à un milliard de nombres, les boucles « for » s'avèrent être deux fois plus rapides que les boucles « BOOST\_FOREACH ». J'ai donc modifié toutes les boucles de Boost présentes dans le code source en les remplaçant par des simples boucles « for ».

```
704
705 int i = 0;
706 BOOST_FOREACH(DISPCurvePtr curve, curveListMem)
707 {
708     i = 0;
709     //On regarde dans quel matrix, on va travailler.
710     //presentStructList.reInit();
711     while((strElt = presentStructList.getCurrent()) != NULL)
712     BOOST_FOREACH(string & strElt, presentStructList)
713     {
714         if(strElt == curve->getStringId().c_str())
715             break;
716         ++i;
717     }
718 }
719
```

Figure 17. Remplacement des boucles BOOST\_FOREACH

## Le changement du code pour s'adapter à la librairie standard C++

Boost est une librairie précurseur de la librairie standard : de plus en plus de fonctions qui étaient à l'origine présentes dans Boost ont été assimilées à la std (librairie standard). Ces fonctions ont été modifiées et ont été rendues plus fiables et plus puissantes. C'est pourquoi j'ai modifié certaines fonctions anciennement utilisées par Boost qui font aujourd'hui partie de la librairie standard afin de garantir plus de stabilité dans le code.



```
395 boost::shared_ptr<TTransport> socket(new TSocket(serverInfo.ipAddress, serverInfo.port));
396 boost::shared_ptr<TTransport> transport(new TBufferedTransport(socket));
397 boost::shared_ptr<TProtocol> protocol(new TBinaryProtocol(transport));
398
399
400 std::shared_ptr<TTransport> socket(new TSocket(serverInfo.ipAddress, serverInfo.port));
401 std::shared_ptr<TTransport> transport(new TBufferedTransport(socket));
402 std::shared_ptr<TProtocol> protocol(new TBinaryProtocol(transport));
403
```

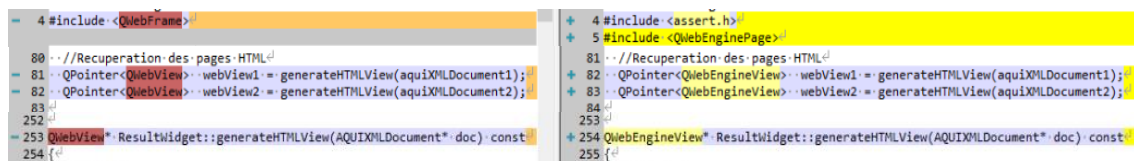
Figure 18. Changement de "boost::shared\_ptr" en "std::shared\_ptr"

## Les changements dans le code pour correspondre aux nouvelles librairies.

À chaque fois qu'une librairie a été mise à jour, il faut ensuite modifier le code pour qu'il puisse compiler.

### Prenons l'exemple de la librairie Qt :

Dans Qt, beaucoup de variables et données ont été modifiées, réorganisées ou même supprimées. J'ai donc établi la liste de ces dernières en notant les nouvelles à utiliser.



```
4 #include <QWebFrame>
80 //Recuperation des pages HTML
81 QPointer<QWebView> webView1 = generateHTMLView(aquiXMLDocument1);
82 QPointer<QWebView> webView2 = generateHTMLView(aquiXMLDocument2);
252
253 QWebView* ResultWidget::generateHTMLView(AQUIXMLDocument* doc) const
254 {
255
256 #include <assert.h>
257 #include <QWebEnginePage>
81 //Recuperation des pages HTML
82 QPointer<QWebEngineView> webView1 = generateHTMLView(aquiXMLDocument1);
83 QPointer<QWebEngineView> webView2 = generateHTMLView(aquiXMLDocument2);
84
254 QWebEngineView* ResultWidget::generateHTMLView(AQUIXMLDocument* doc) const
255 {
```

Figure 19. Changement du code source pour s'adapter à la librairie Qt

Comme illustré sur la Figure 19, j'ai procédé à des changements nécessaires pour assurer la compatibilité de la nouvelle version de la librairie Qt 5.15.1.

Jusque-là, j'ai donc modifié les scripts python, mis à jour les librairies et j'ai adapté le code source d'ARTISCAN à ces nouvelles librairies. Pour vérifier que ces changements sont corrects, il faut valider deux étapes : la **compilation** et l'**exécution**.

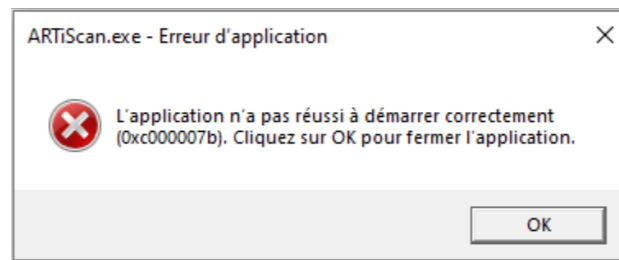
Tous les projets d'ARTISCAN compilent, ce qui veut dire que tous les exécutables sont bien générés.

ARTIScan.exe	21/12/2021 16:23	Application	7 534 Ko
ASCKeygenerator.exe	16/12/2021 15:13	Application	184 Ko
ASCMigrationTools.exe	16/12/2021 15:08	Application	2 192 Ko
ASCServer.exe	16/12/2021 12:01	Application	38 Ko
ASCServerManager.exe	21/12/2021 16:19	Application	3 380 Ko
CipherTool.exe	21/12/2021 16:18	Application	219 Ko
ProjectArtiscanValidation.exe	16/12/2021 15:09	Application	687 Ko

Figure 20. Les exécutables d'ARTISCAN sont générés



Néanmoins, dès lors qu'on veut lancer l'une des applications de la *Figure 20*, l'erreur suivante apparaît :



*Figure 21. Erreur d'application au lancement de ARTIScan.exe*

Cette erreur signifie que l'application a planté. Ce plantage peut provenir de plusieurs sources, c'est pourquoi dans le but d'identifier le problème, j'ai eu l'idée de créer un mockup de l'application ARTISCAN.

## C. Tâches périphériques

### 1. Mockup Artiscan

Afin de d'identifier le problème de l'exécution au moment du lancement de l'application, j'ai réalisé un mockup de l'application ARTISCAN.

#### **Qu'est-ce qu'un mockup ?**

- Un mockup fait référence à une maquette. Sa mission principale est de simuler les aspects du logiciel en allégeant ses fonctionnalités. Il est aussi employé pour anticiper et estimer la complexité d'un projet.

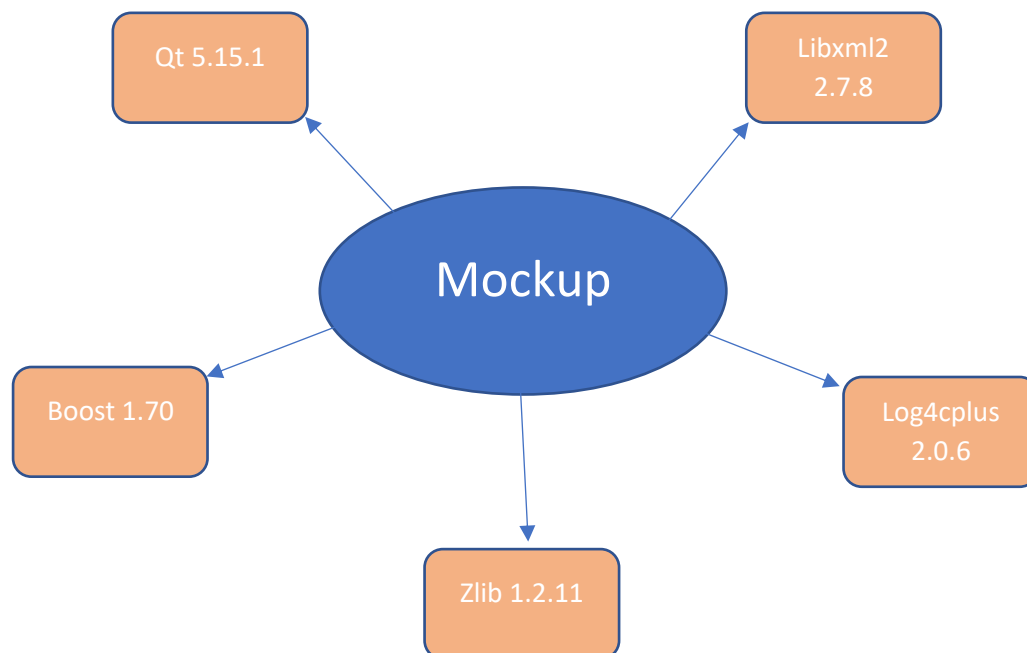
Dans le cadre d'ARTISCAN, le mockup permettra :

- D'identifier le/les plantages dans les exécutables
- De vérifier que les nouvelles librairies sont fonctionnelles
- De vérifier que le code source renvoie les valeurs attendues

Ce mockup est un projet inclus dans la solution ARTISCAN. Il fait partie des 48 autres que composent l'application et il est directement intégré dans l'environnement de développement de VS2019.

La vérification des nouvelles librairies prend la forme de plusieurs fonctions qui vont chacune appeler des fonctions principales des librairies. Si l'appel de ces fonctions ne provoque pas d'erreur, alors on pourra valider l'utilisation de la librairie correspondante. Sinon, la librairie n'est peut-être pas bien liée au projet (fichiers d'en-tête ou fichiers .dll manquants), ou bien il peut y avoir une incompatibilité entre la librairie et le projet (architecture 32bits ou 64bits).

Dans le mockup, j'ai ainsi testé et vérifié les librairies suivantes :



Ces vérifications ont permis de corriger l'incompatibilité d'une version de la librairie libxml2 avec ARTISCAN. En effet, celle que j'avais initialement choisie était compilée en 64bits.

Comme la solution ARTISCAN est en 32bits, l'application plantait au démarrage mais après l'avoir changé à nouveau, les exécutables de la *Figure 20* s'exécutent sans afficher le message d'erreur de la *Figure 21*.

Cela m'a aussi permis de détecter que la librairie Thrift utilisée pour la partie serveur d'ARTISCAN ne fonctionne pas du tout. Sa remise à niveau est très compliquée en raison de son manque de documentation.

## IV. BILAN

## A. Les résultats

### **Les scripts python :**

J'ai modifié tous les scripts python appelant Visual Studio 2010 pour qu'ils s'adaptent à Visual Studio 2019.

### **Les librairies**

Les librairies mises à jour sont fonctionnelles, elles sont reconnues par Visual Studio 2019 et sont utilisables dans le code. Une librairie importante pour ARTISCAN : Thrift, qui est utilisée pour la partie serveur de l'application, est très compliquée à corriger. Actuellement, c'est la seule librairie qui n'est pas fonctionnelle parmi toutes celles du logiciel.

### **Le code source**

Depuis une incompatibilité totale du code dans l'environnement de développement, j'ai réussi à faire en sorte que le code source soit compatible avec la norme C++14 tout en respectant l'utilisation des nouvelles librairies.

### **La compilation des projets**

Tous les projets qui génèrent les exécutables d'ARTISCAN compilent sous Visual Studio 2019.

### **L'exécution des applications**

Grâce à l'utilisation du mockup, j'ai identifié des librairies à modifier. Maintenant, les applications ne plantent plus au démarrage mais elles ne présentent pas le même comportement que sur l'ancienne version d'ARTISCAN (celle fournie au début de mon stage). Certaines interfaces d'utilisateur ne sont pas affichées en raison d'erreurs encore existantes dans le code ainsi que la non-conformité de la librairie Thrift.

## B. La suite de mon travail ?

J'ai rédigé un compte-rendu des démarches effectuées pendant toute la durée de mon stage, qui contient toutes les corrections du code ainsi que toutes les corrections restantes à faire. Dans ce compte-rendu, j'ai détaillé la mise en place du poste de développement, où retrouver mon travail et comment le poursuivre dans les meilleures conditions possibles.

Mon conseil pour la suite de ce travail pourrait être de solliciter un développeur dit « sénior » afin qu'il puisse résoudre les problèmes persistants sur la partie serveur d'ARTISCAN. Ce déblocage permettrait aux développeurs actuels de l'équipe de pouvoir continuer à régler les problèmes d'exécution et ainsi pouvoir définitivement porter l'application sur Visual Studio 2019.

Je pense que la correction des dernières erreurs à l'exécution serait un bon sujet de stage

Grâce à cela, la dette technique qui pèse sur ARTISCAN et l'empêche d'intégrer de nouvelles fonctionnalités très puissantes pourrait être rattrapée.

## Conclusion

Ma mission au cours de ces derniers mois était de mettre à niveau la solution ARTISCAN de Visual Studio 2010 vers 2019.

Pour répondre à ce besoin, j'ai tout d'abord modifié les scripts python pour les adapter au nouvel environnement de développement. Ensuite, dans la recherche de performances, j'ai mis à jour les libraires externes et adapté le code source à celles-ci. Enfin, pour en vérifier leur cohérence avec Visual Studio 2019, j'ai mis en place un mockup qui m'a permis de corriger certaines librairies. De cette manière, j'ai ainsi pu lancer les exécutables générés lors de la compilation.

Ce stage m'a permis de développer ma rigueur, mon sens du détail ainsi que mon esprit de curiosité et d'adaptation. C'est non seulement une expérience très enrichissante d'un point de vue personnel, mais aussi un grand apport professionnel.

Pour mes prochains stages, je pense néanmoins étudier plus en détail les missions que l'on me proposera. D'une part pour avoir plus de renseignements et d'avoir un avis plus précis sur mon travail, mais aussi pour comprendre les raisons qui ont poussé à la réalisation de ce stage. Je pense qu'il me manquait un gros bagage technique pour pouvoir réaliser les missions complètement et pouvoir rendre un travail fonctionnel. De plus, même si l'équipe de développement m'a bien encadré et aidé, je regrette un peu d'avoir effectué un travail indépendant de l'équipe et de ne pas avoir eu à effectuer un travail collaboratif.

En tant que premier stage, je garde cependant un très bon souvenir de l'environnement de travail. Cette expérience professionnelle constitue un atout et un véritable socle de connaissances sur lesquelles je pourrai m'appuyer pour la fin de mes études et pour mes prochaines expériences professionnelles.

## Bibliographie

- <https://www.e-cancer.fr/Patients-et-proches/Se-faire-soigner/Traitements/Radiotherapie>
- <https://www.ionos.fr/digitalguide/sites-internet/developpement-web/explication-de-la-dette-technique/>
- <https://www.adimeo.com/blog/dette-technique>
- <https://www.aquilab.com/fr/accueil/>
- <https://www.linkedin.com/company/aquilab/about/>
- <https://www.techno-science.net/definition/5358.html>
- [https://fr.wikipedia.org/wiki/Microsoft\\_Visual\\_Studio](https://fr.wikipedia.org/wiki/Microsoft_Visual_Studio)
- <https://www.boost.org/>
- <https://www.qt.io/>
- <https://docs.microsoft.com/fr-fr/visualstudio/releases/2019/compatibility>
- <https://docs.microsoft.com/fr-fr/cpp/windows/latest-supported-vc-redist?view=msvc-170#visual-studio-2010-vc-100-sp1-no-longer-supported>
- [https://fr.wikipedia.org/wiki/Visual\\_C%2B%2B](https://fr.wikipedia.org/wiki/Visual_C%2B%2B)
- <https://isocpp.org/std/the-standard>
- <https://linuxfr.org/news/les-coulisses-du-standard-cpp#les-versions-c>
- <https://levelup.gitconnected.com/the-definite-guide-on-compiling-and-linking-boost-c-libraries-for-visual-studio-projects-c79464d7282d>
- <https://fr.wikipedia.org/wiki/CMake>
- [https://www.siliceum.com/fr/blog/post/cmake\\_01\\_cmake-basics/](https://www.siliceum.com/fr/blog/post/cmake_01_cmake-basics/)
- <https://www.usabilis.com/definition-mockup/>

## Annexe :

