**TRAVAUX / MISSIONS**

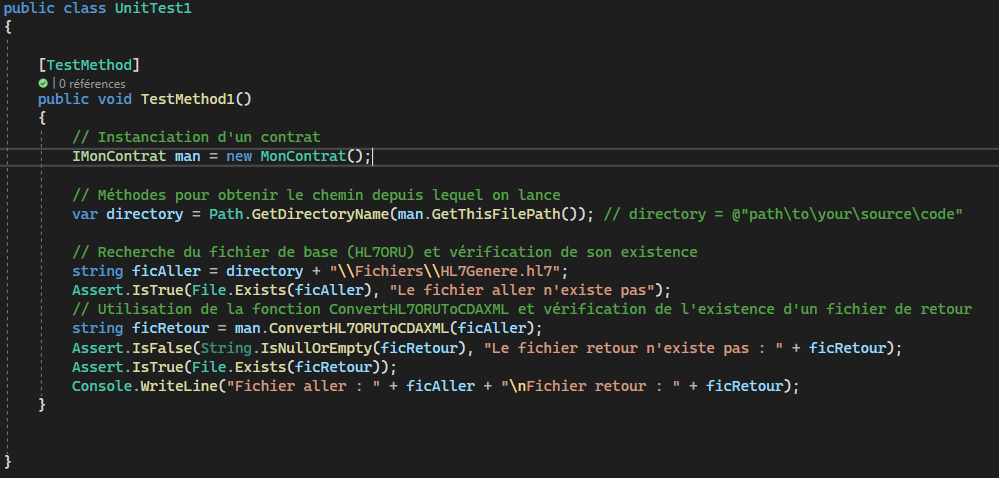
**Jour 1 -> Jour 5**

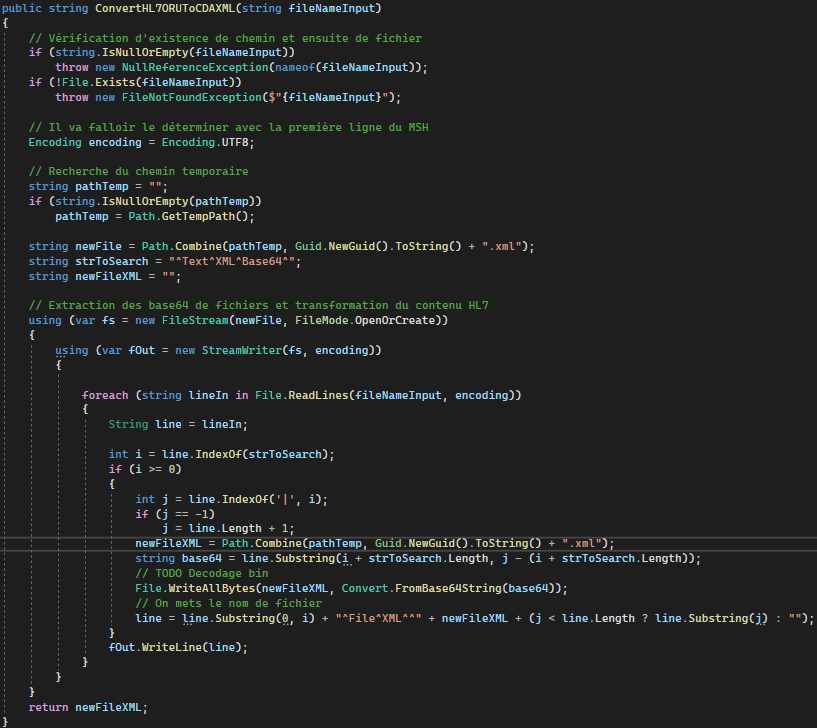
* Apprentissage C# pour être prêt et à l’aise lors du développement
  + Cours Berger-Levrault
  + Site dotnetcademy.net
  + Exercices trouvés sur Internet (morpion, pendu, gestion de parc...)
  + Création d’un jeu de MONOPOLY

**Jour 6 -> Jour 10**

Ce projet a pour finalité de produire une page web permettant de convertir un fichier HL7 en fichier XML tout en récupérant (s’il y a présence) d’un PDF. Le programme va aussi générer des fichiers anonymes (Puisqu’on se trouve dans un cadre médical avec beaucoup d’informations personnelles)

* Développement en C# depuis VS, d’un convertisseur de fichier HL7ORU to XML (2h)



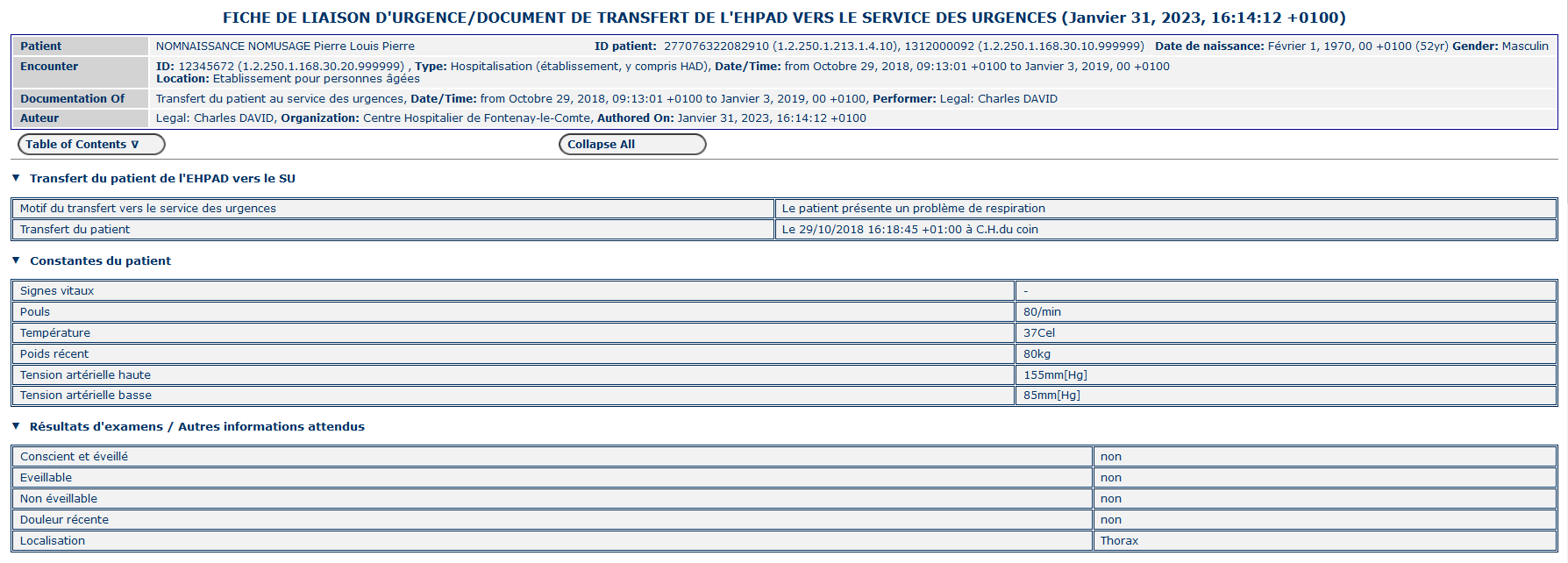




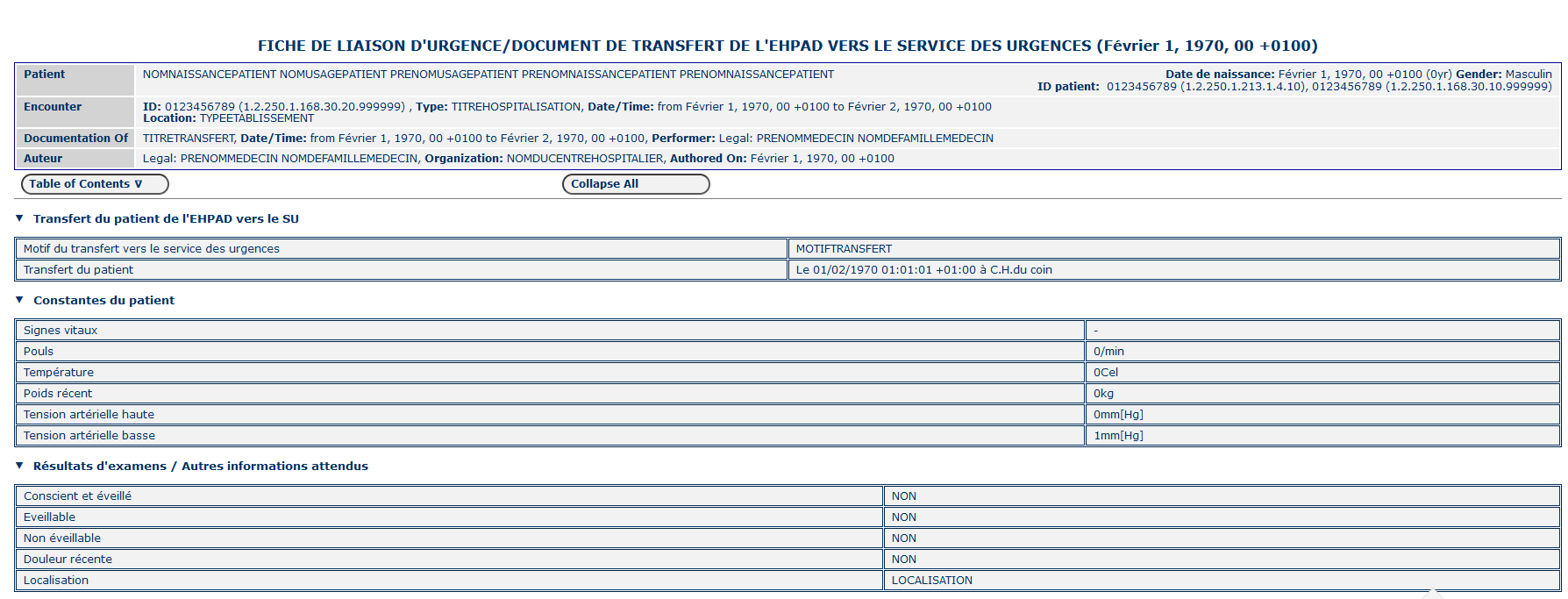


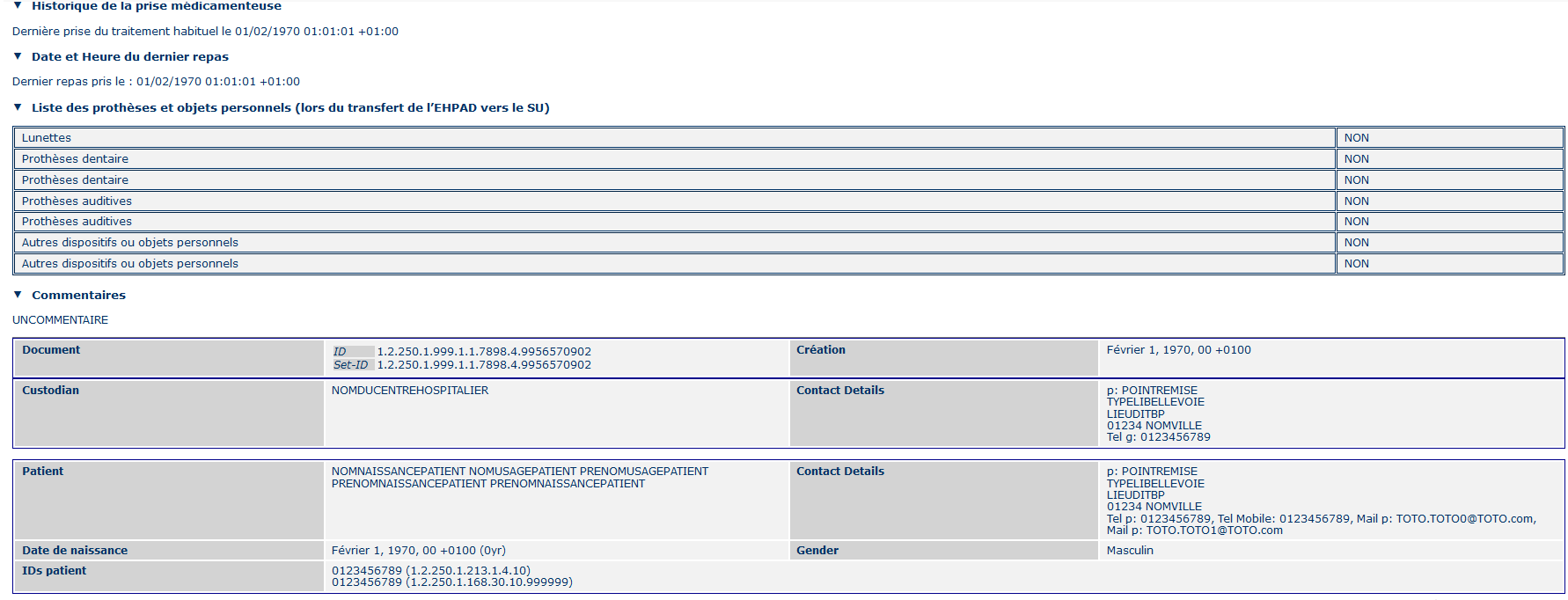
* Documentation sur le CDA (Clinical Document Architecture) (30min)
* Modification du fichier xml depuis une fonction C# pour anonymiser des noms, des identifiants...

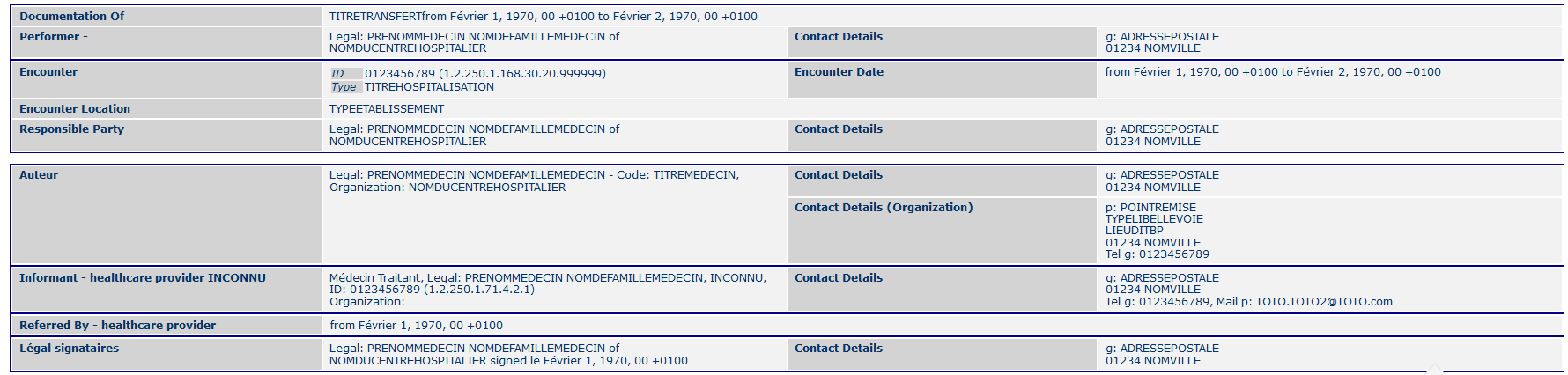
On part d’un fichier quelconque (Feuille de style activée) :



Pour passer à un fichier anonyme (Feuille de style activée) :







* CDA avec un PDF (besoin de l’extraire)
  + Tout d’abord, il faut convertir le fichier hl7 de base en XML (en récupérant toute la base64)
  + Ensuite, il faut le récupérer et après la vérification de la présence d’un PDF à l’intérieur, on peut extraire la base64 du XML
  + Enfin, on peut procéder à la conversion
* Ajustement du programme pour prendre en compte les modifications d’un CDA de niveau 3 (auto présentable) contenant les mêmes données mais agencées différemment)
* Dans le domaine médical, il y a des normes et les fichiers doivent subir une validation
* En attente de la vérification du fichier de niveau 3 anonymiser, j’ai cherché s’il était possible d’automatiser la validation directement lorsque l’on injecte un fichier hl7

(La capture d’écran montre qu’il est possible de récupérer le résultat de la validation, néanmoins, je ne sais pas encore si c’est exploitable et utilisable ou pas)

* Le projet se concrétise et il faut maintenant créer une page WEB ayant pour but de réellement faire marcher la conversion. Après une journée de travail, j’arrive à une page fonctionnelle (voir ci-dessous)



* Le principe est simple :
  + On met le fichier hl7 que l’on souhaite et on l’upload (Mise en ligne) sur la page.
  + La page ayant le fichier, va réaliser toutes les fonctions de conversion et de récupération PDF pour ensuite le mettre à disposition au téléchargement en dessous
  + Note : avant et après chaque utilisation, le site vide les fichiers qu’il a reçus.

**Jour 10 -> Jour 27**

Restructuration du projet : Abandon de l’idée d’implémenter une page à un site web existant, à la place, une nouvelle page sera disponible, elle va elle utiliser de l’asp.net core (Besoin de documentation sur les nouveau Tips de langages) Il y a aussi les dockers (conteneurs) à comprendre et à savoir utiliser

* Ajout d’un nouvel outil d’export de fichier CDA (en .xml) à partir de zip IHEXDM
* Le projet ici est de créer une image docker de la solution pour qu’elle puisse être utilisable très facilement (Avec la documentation ci-dessous, il est possible d’accéder au site fonctionnel en local très facilement)
* **Documentation :**

1. Obtenir la Solution bl.sante.interop.microservices
2. Ouvrir l’invite de commandes en mode administrateur
3. Vérifier si votre ordinateur dispose d'une installation de Linux. Si ce n'est pas le cas, vous pouvez installer Windows Subsystem for Linux  
   (Commande pour installation : « wsl –install »)
   1. Pour plus d’informations : https://learn.microsoft.com/en-us/windows/wsl/install
4. Pour rentrer sur la partie Linux si vous n’y êtes pas encore, entrer :
   1. « bash » ou « wsl »
5. Installer docker en entrant :
   1. « sudo apt-get update && sudo apt-get install -y docker.io »
6. Se positionner dans le répertoire bl.sante.interop.microservices   
   (via la commande « cd NomDuDossier» jusqu’au répertoire de la solution «  cd bl.sante.interop.microservices »)
7. Entrer :
   1. « docker pull tomyrulliat/outildeconversion »
   2. « docker build -t tomyrulliat/outildeconversion -f ./Conversion\_Tool/Dockerfile . » (le point à la fin est important !)
   3. « docker run -d -p 8080:80 --name OutilDeConversion tomyrulliat/outildeconversion »
8. Aller sur le navigateur puis aller sur : <http://localhost:8080/>

* **Help :** 
  + See all containers
    - « docker ps »
  + See all docker images
    - « docker images »
  + Uninstall docker then install docker again
    - « sudo apt-get remove docker docker-engine docker.io »
  + Stop, start and restart the docker via terminal
    - « docker stop <CONTAINER ID> »
    - « docker start <CONTAINER ID> »
    - « docker restart <CONTAINER ID> »
  + Delete the container and his image
    - « docker stop OutilDeConversion && docker rm OutilDeConversion && docker rmi tomyrulliat/outildeconversion »
* Ajout de la fonctionnalité multi-langage et d’un manuel d’utilisation (voir les captures d’écran du jeudi 15/06/2023
* Travail sur le design et l’ergonomie de la page (couleur et taille des boutons, textes en gras…) 🡪 Besoin de correspondance à une charte graphique.

**Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement**

* Fonctionnalité de vérification des documents CDA à chaque envoi de document HL7ORU ou IHE\_XDM
* Code propre avec commentaires explicatifs

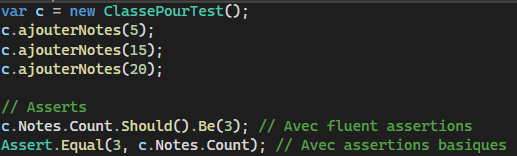
**JOURNAL DE BORD**

**Jour 1 (15 mai 2023)**

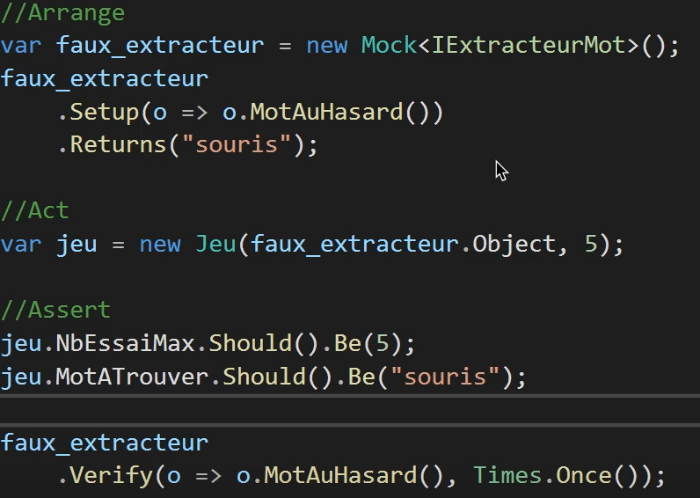
* Découverte des lieux
* Rencontre avec l’équipe
  + Simon PIJARD (Maître de stage en 5ème année de master)
  + Leo SOHRABI (Alternant en fin de 2ème année de BTS SIO)
  + Phillipe LUCIANI (Ingénieur Support et développement)
  + Lucas DAMIAN-PICOLLET (Alternant en fin de 2ème année)
  + Emmanuelle ELUMBA (Analyste Tests et Validation)
  + Laureline BEAL (Ingénieure en développement)
  + Lucy CADIOU (Consultante avant-vente)
  + Cédric MAFREYT (Ingénieur Support et développement)
* Récupération et installation des équipements :
  + Ordinateur dans le “Flex-Office”
  + Tout l’environnement d’après un PDF de l’entreprise : VS Code (et toutes les extensions utiles pour toute la continuité du stage : connexion à git, Team Explorer, Silverlight...)
* Cours de C# : apprentissage de 13h à 17h (interne au groupe Berger-Levrault et similaire aux cours open-Classroom)
  + Révisions des bases de C# (string, int, var, const...)
  + Application d’algorithme (Pendu)
  + Notions d’héritages
  + Try/Catch
  + Découverte des tests sur Visual Studio
  + Temps de documentation C# obligatoire pour faciliter la compréhension (≈30 min)

**Jour 2 (16 mai 2023)**

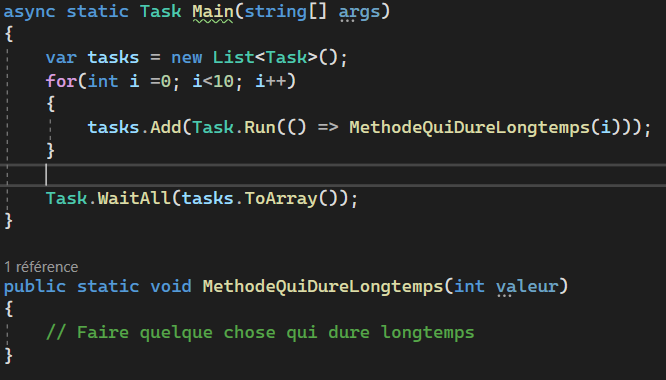
* Cours de C# avancé de 9h à 17h :
  + Tests avec le package fluent assertions :



* + Les mock avec les setups, returns et verify :



* + La redimension de tableau avec les méthodes :
    - Array.Resize(ref monTableauARedimensionner, NouvelleValeur) ;
    - La création d’une liste et le passage ensuite en tableau (plus flexible et moins coûteux
      * MaListe.ToArray() ;
  + Collections algorithmiques :
    - Piles = Stack (Push / Pop)
    - Files = Queue (Enqueue / Dequeue)
  + LINQ (Logique SQL dans C#)
    - Voir programme LINQ
  + Précision de l’héritage
    - Interdiction d’héritage : sealed
    - Possibilité de modifier des champs dans la classe fille : new
    - Redéfinition d’opérateurs
    - Redéfinition des méthodes objets
      * ToString()
      * Equals(obj) --> Obligation d’une vérification du type :
        + If (obj is string) {…}
      * GetHashCode()
  + Découverte des évènements en C#
  + Asynchronisme = arrière-plan (lancer quelque chose et pouvoir faire quelque chose en même temps => lancer une tâche en fond et faire quelque chose en attendant qu’elle soit terminée)
    - Async
    - Await



* + Formation autonome sur le Site dotnetcademy.net (sur les queues et les stacks)
  + LinkedLists
  + Codage de plusieurs algorithmes dont un morpion contre un ordinateur choisissant des places aléatoires.
  + Temps de documentation C# obligatoire pour faciliter la compréhension (≈1h)

**Jour 3 (17 mai 2023)**

* Visionnage du projet de mon maître de stage (≈ 1h)
  + Architecture se rapprochant du MVC (que je vais utiliser prochainement)
    - Managers (et leurs contracts)
    - UseCases
    - Models
    - Des tests
  + Back-office pour la gestion des clients
  + Utilisation HTML, CSS, JSON (TypeScript), C# avec nottement LINQ permettant de réaliser des requêtes
* Création d’un programme de gestion de parc de véhicules en C# (≈2h)
  + Objets véhicule
    - Objets “voiture” héritant de véhicule
    - Objets “utilitaire” héritant de véhicule
  + Des méthodes de gestion : ajout, suppression
  + Des méthodes d’affichage
* Documentation JSON pour comprendre son fonctionnement (≈1h)
* Démarrage d’une activité C# utilisant l’objet et l’héritage : MONOPOLY GAME

**Jour 4 (22 mai 2023)**

* Finalisation du projet MONOPOLY GAME (voir ci-dessous)





Toutes les règles du jeu de base sont prises en compte :

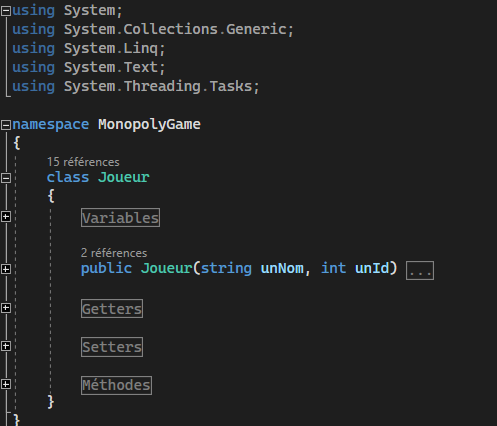
* Case prison
* Les services (nombre de dés \* 10)
* Lorsque l’on obtient toutes les cartes d’une même couleur, leur loyer double
* Les cartes chance et communauté sont prises en compte dans le jeu
* Les ordinateurs déterminent aléatoirement leur choix
* Loyers, achats, mouvements sont pris en compte
* Des affichages permettent une excellente compréhension lors du jeu
* La fin de partie suite à une vérification d’une faillite d’un des joueurs

Plusieurs règles ne sont pas prises en compte :

* Les maisons et hôtels

**Jour 5 (23 mai 2023)**

* Propreté du code MONOPOLY :



* Ajout des fonctionnalités suivantes :
  + Plus d’affichages : cartes, plateau...
  + Possibilité de rachat de carte
  + Propreté dans la console
  + Possibilité d’achat de maisons et d'hôtels
* Recherche de bugs dans le jeu ensuite résolu
* Grande phase de test pour trouver tous les problèmes rencontrés
* Exercices CSS rapides : <https://www.w3schools.com/css/exercise.asp?filename=exercise_padding2> (≈1h)

**Jour 6 (24 mai 2023)**

* Finalisation de mise en place de l’environnement de travail (Connexion à TFS : “Team Foundation Server” pour la gestion des sources et des builds, le suivi des éléments de travail, la planification, la gestion de projet et l'analyse des performances)
* Démarrage du projet principal

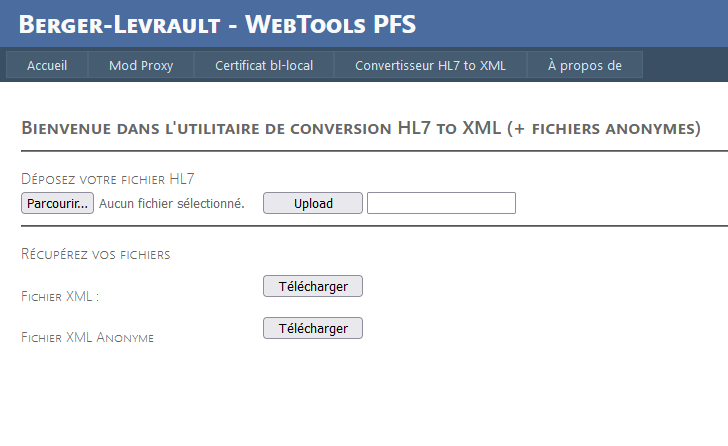
**Jour 7 (25 mai 2023)**

* Anonymisation du document XML régulé (trop de choses étaient cachés)
* Création d’un convertisseur lorsqu’un document PDF au format base64 se trouve dans un fichier XML
* A l’heure actuelle, le programme de convertisseur permet de rentrer un document de type hl7 et de ressortir un fichier xml et son PDF mais aussi un fichier xml anonymiser ainsi qu’un PDF vierge

**Jour 8 (26 mai 2023)**

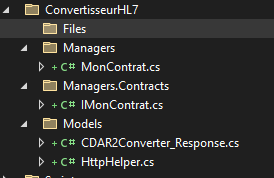
* Découverte de l’existence des plusieurs niveaux de CDA (le niveau 1 représente un fichier xml avec les informations principales et un PDF avec toutes les informations)
* Ici, j’utilise un CDA de niveau 3 qui est auto présentable (il contient notamment du CSS et beaucoup d’autres balises)
* Dans le domaine médical, il y a des normes et les fichiers doivent subir une validation
* En attente de la vérification du fichier de niveau 3 anonymiser, j’ai cherché s’il était possible d’automatiser la validation directement lorsque l’on injecte un fichier hl7
* Démarrage de la partie WEB (voir ci-dessous)



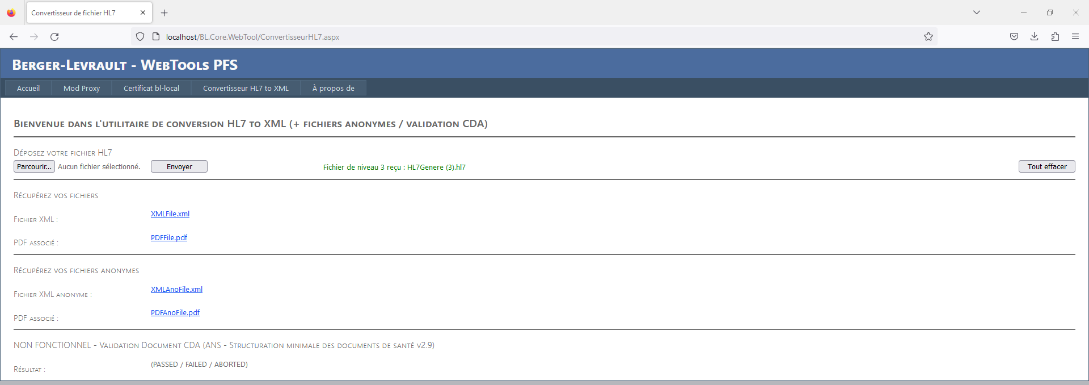


**Jour 9 (30 mai 2023)**

* Réorganisation de mon travail



* Amélioration du programme, et mise en place de toute l’infrastructure pour le bon fonctionnement du site
* Après une journée, j’en arrive à ce résultat :

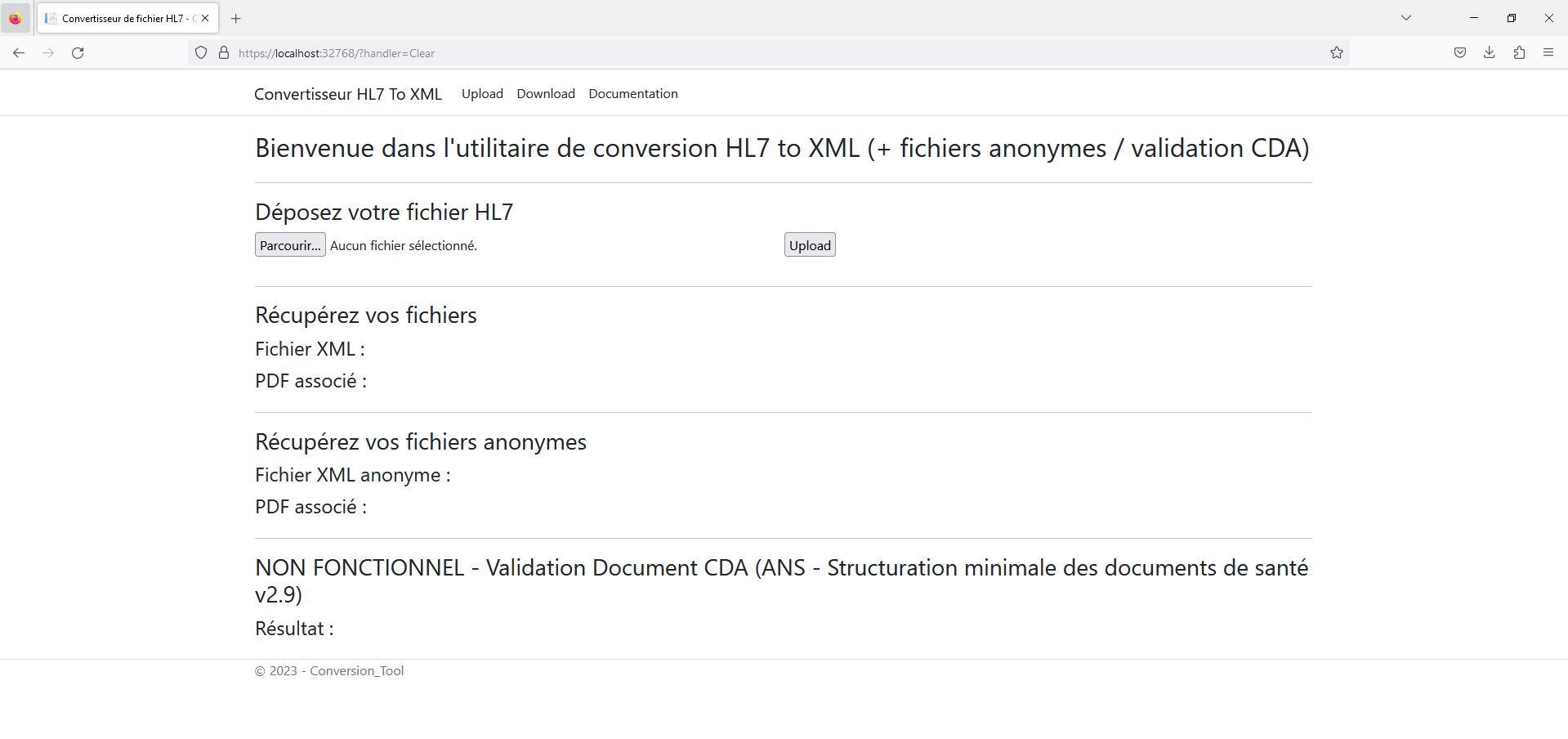


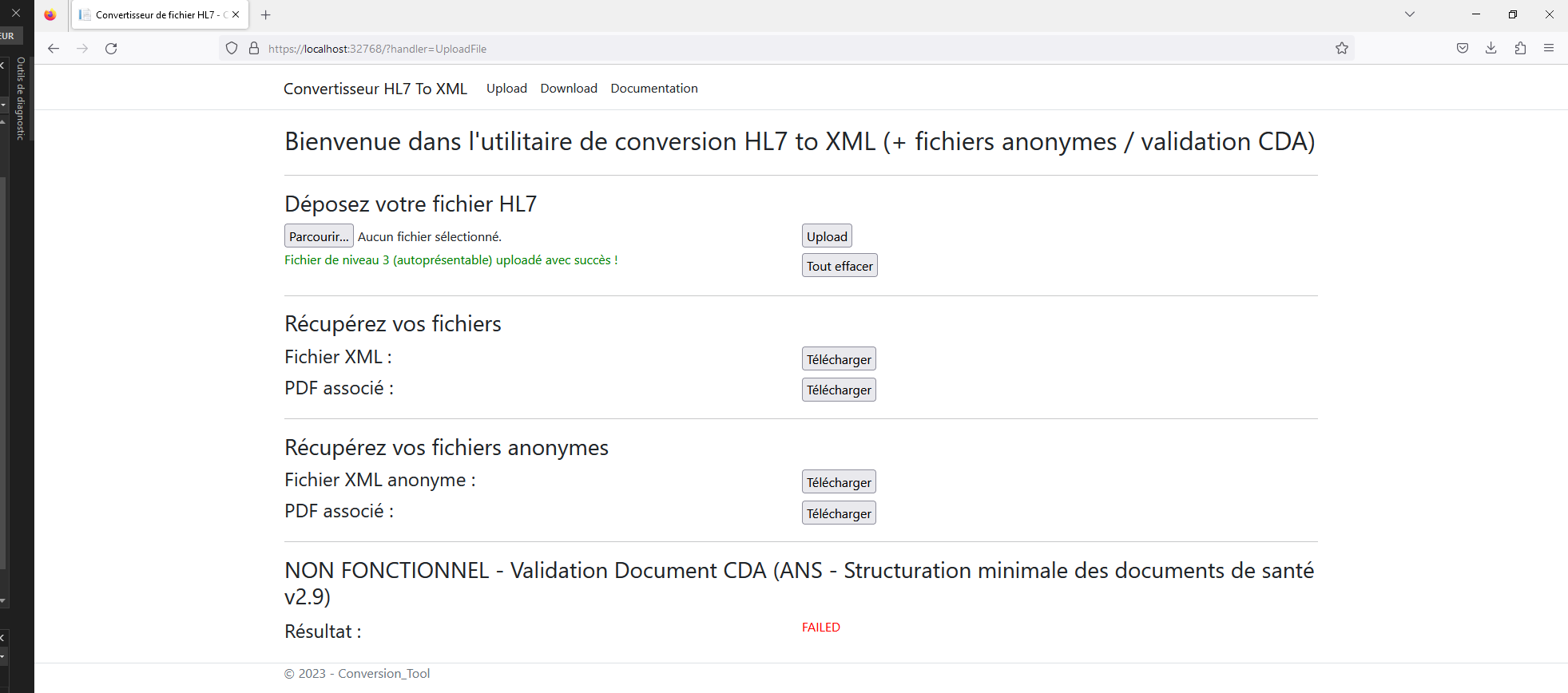
**Jour 10 (31 mai 2023)**

* Veille sur ASPNET (comme je travaille dessus et ayant trouvé des articles datant des années 2000, j’ai voulu me renseigner sur ce Framework)
  + Lancement 5 janvier 2002 par Microsoft
  + ASP.NET est un Framework permettant de générer à la demande des pages web (HTML, CSS, JS, API web...)
* ASP.NET Core est un Framework Web gratuit et open-source, développé par Microsoft et la communauté et sortie initialement le 7 juin 2016. Il est plus performant qu'ASP.NET. C'est un Framework modulaire qui fonctionne à la fois avec le Framework .NET, sous Windows et .NET en multiplateforme
  + Il est plus performant qu'ASP.NET. C'est un Framework modulaire qui fonctionne à la fois avec le Framework .NET, sous Windows et .NET en multiplateforme.
  + Plusieurs mises à jour sont effectuées régulièrement
* Dans mon cas, je travaille sur de l’ASP.NET
* Discussion avec Philippe : Restructuration du projet (voir au-dessus)
* Mise en place des logiciels :
  + WSL (On essaye d’utiliser DOCKER sauf qu’étant payant sur WINDOWS, il nous faudra l’utiliser sur une interface LINUX)
  + Docker Desktop
  + Travail sur l’invité de commande pour permettre la connexion
* Comme l’on utiliser de l’ASP.NET Core, il y a un grand temps de documentation et d’apprentissage autonome (environ 3-4h)
* Plusieurs essais, résolution de bugs, redémarrage de l’application...

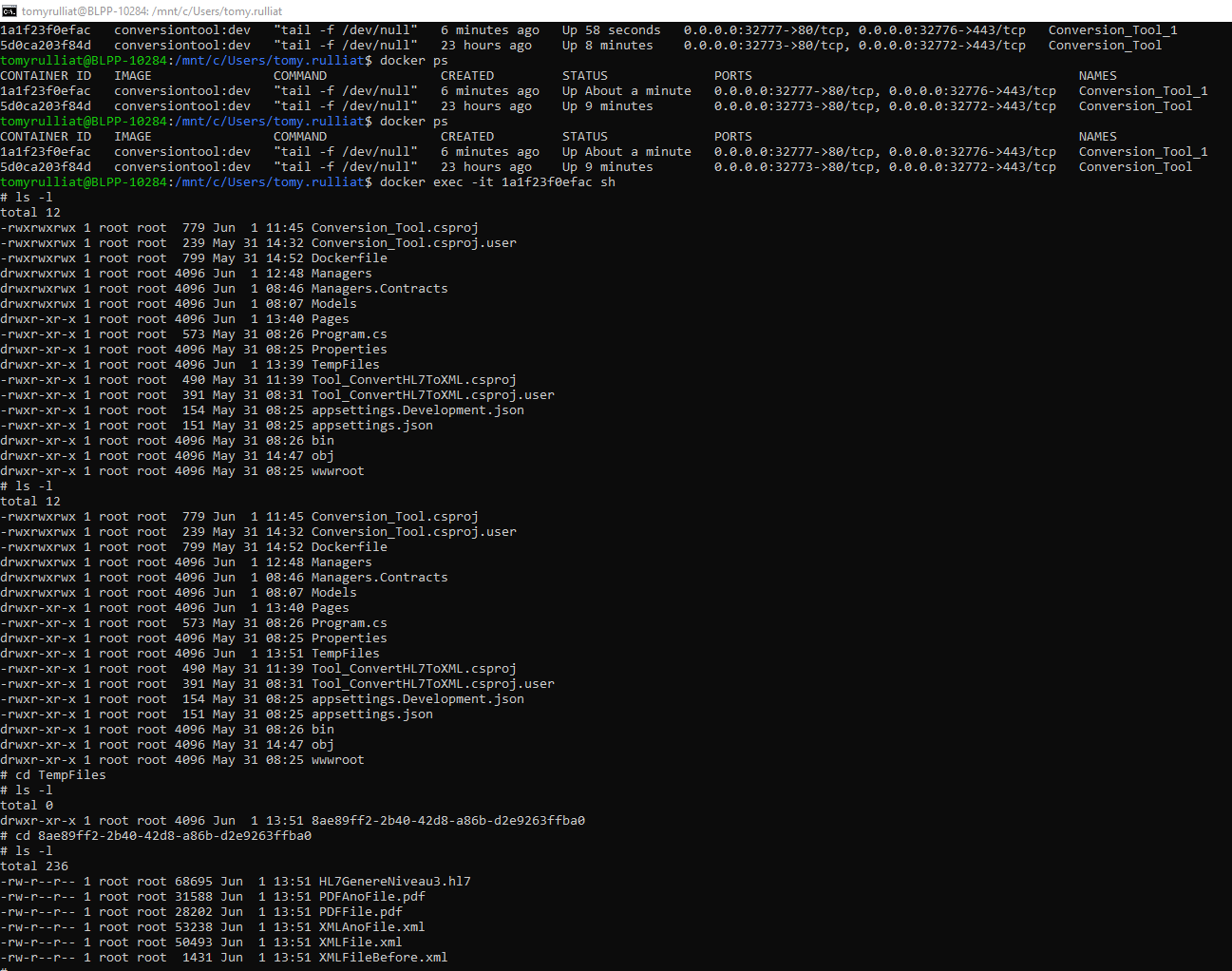
**Jour 11 (1er juin 2023)**

* Prise de temps personnel pour faire un point sur les objectifs (dans un calepin)
* Compréhension de la structure d’une application ASP.net Core (nouvelles balises et nouveaux liens)
* Structuration du site
* Démarrage d’une documentation pour expliquer et faciliter la compréhension de mon projet





Le lien avec WSL (invité de commande BASH – Linux) :



**Jour 12 (2 juin 2023)**

* Correction de noms de variables pour faciliter la compréhension
* Elaboration d’une documentation technique sur l’application web pour permettre une maintenance facile et la compréhension de l’utilisation (voir ci-dessous)
* Ajout d’une fonction ClearAll() se lançant au début de l’application pour permettre de vider les fichiers temporaires
* Pour la prochaine fois :
  + Importation de la classe CDAHelper.cs
    - Fonction Analyse\_CDAIHEXDM
  + Création d’une librairie avec tous les besoins de cette classe (ajout des autres classes...)
  + Outil de conversion CDA To IHEXDM

**Jour 13 (5 juin 2023)**

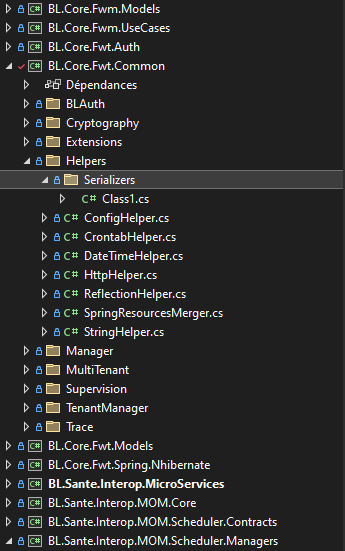
* Changement d’architecture : En voulant utiliser la fonction Analyse\_CDAIHEXDM, je me suis rendu compte qu’il fallait importer et utilisateur plusieurs dizaines d’autres classes et interfaces, on a alors décider de tout recommencer et de me faire travailler en même temps sur le projet de mon maitre de stage (qui utilisait les mêmes bibliothèques internes à Berger-Levrault) en récupérant les méthodes que j’avais besoin.



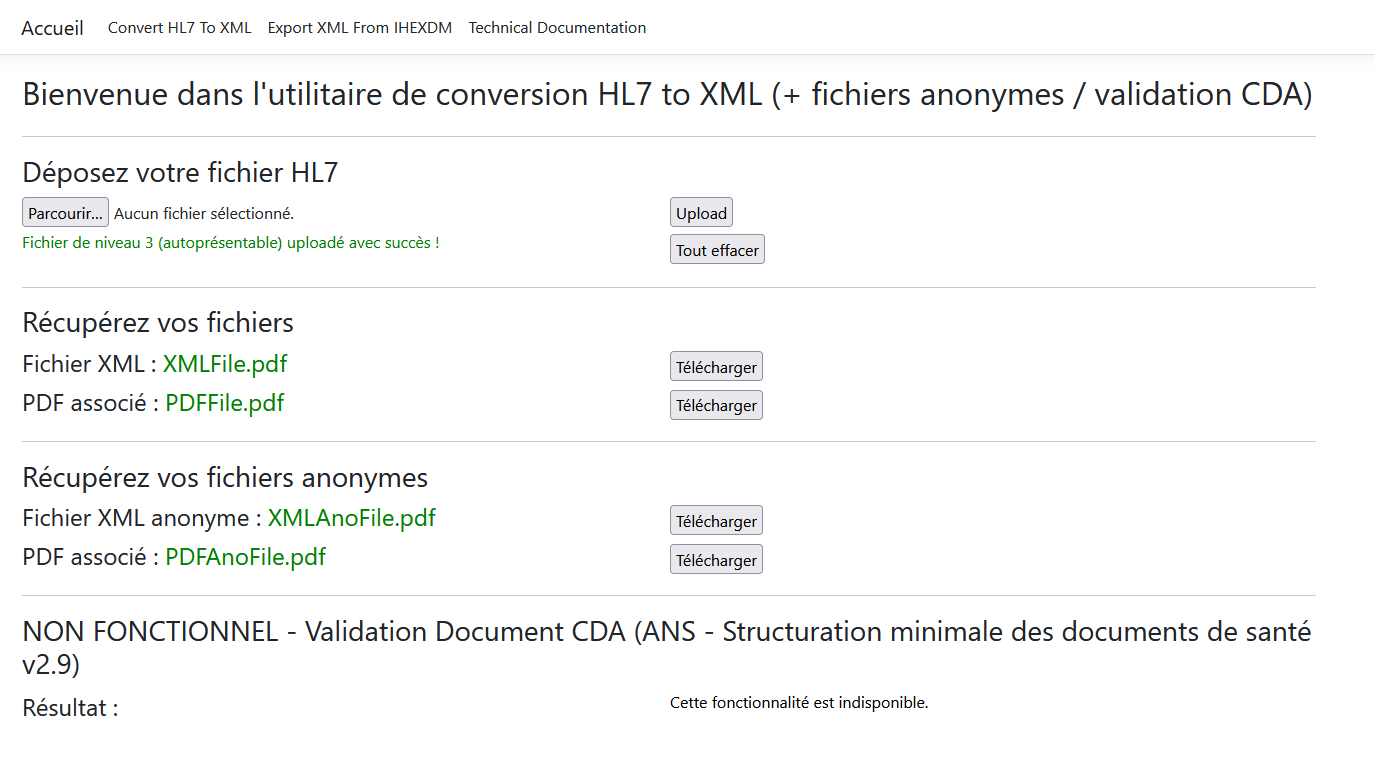
(Dans chacun de ces dossiers se trouvent des plusieurs classes et interfaces)

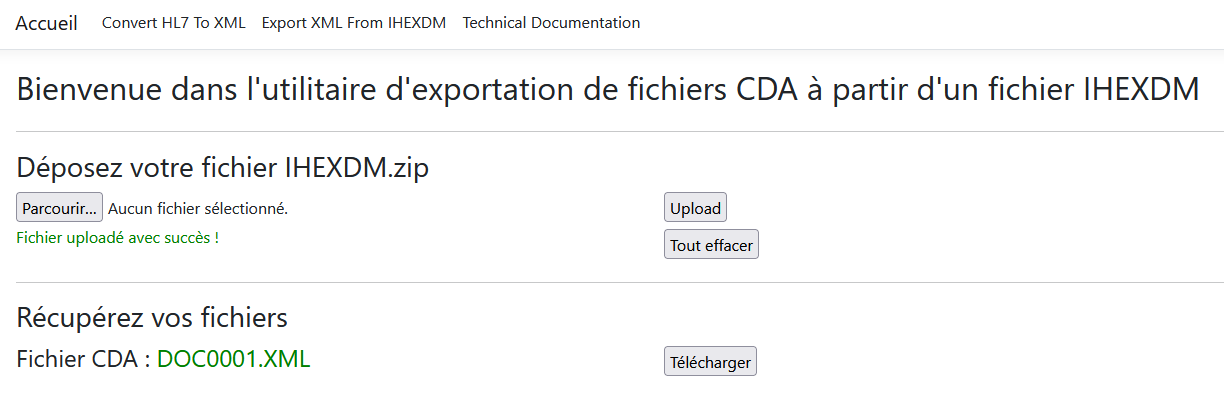
**Jour 14 (6 juin 2023)**

* En vue de l’utilisation d’une fonction propre à Berger-Levrault permettant de sortir les fichiers CDA de fichiers IHEXDM, il y a un besoin d’importer beaucoup d’autres classes et méthodes.
* C’est ainsi qu’il y a eu un grand temps de recherche et d’importation de dossiers pour récupérer tout le nécessaire au besoin de cette fonction.

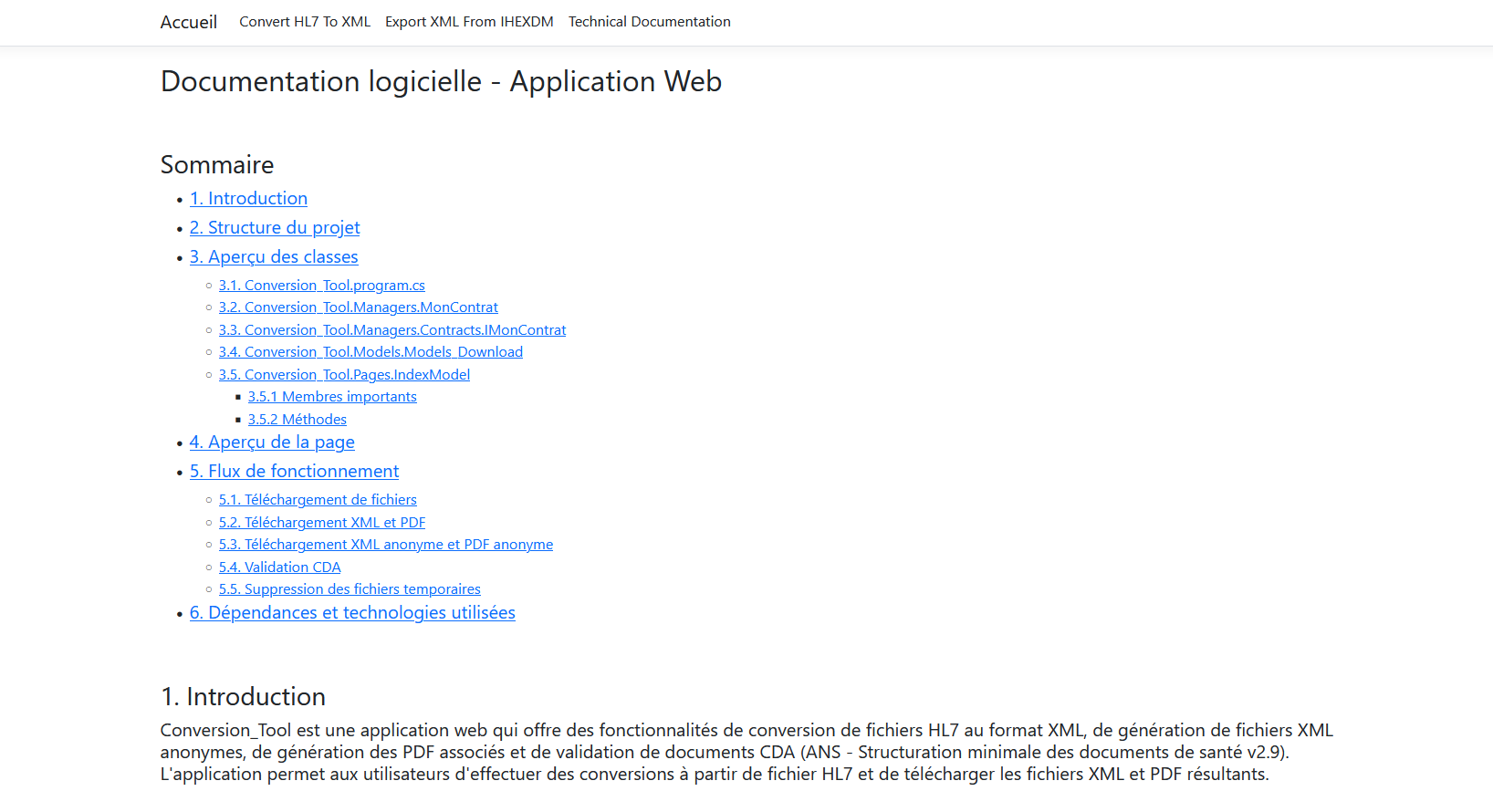


* Après plusieurs bugs résolus, j’en arrive à un site fonctionnel :



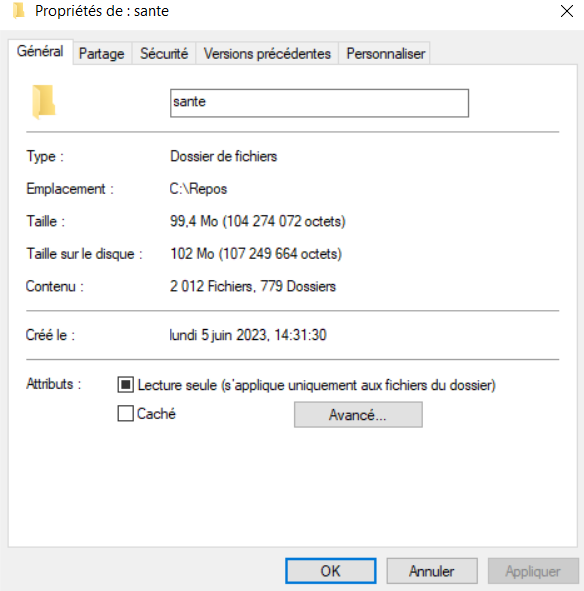


* Documentation sur l’application web (voir ci-dessous)





* Ce qui est impressionnant de noter est le nombre de dossiers / fichiers utilisés alors même que l’habitude était de ne travailler qu’avec 2/3 fichiers



**Jour 15 (7 juin 2023)**

* Mise en place d’injection de dépendances (L’entreprise Berger-Levrault est entrain de basculer de dotnet avec des injections via SPRING à dotnet Core avec des injections via Microsoft)
* Apprentissage et compréhension de l’utilisation des UseCases

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

!!! IMPORTANT !!!

Pour les UseCases :

* Manager
  + UneClasse.cs (héritant de IUneClasse.cs)
    - UneFonction1(string Chaine){ … }
* Manager.Contract
  + IUneClasse.cs (C’est une interface qui permet de se référer aux méthodes de UneClasse.cs)
    - UneFonction1(string Chaine) ;
* Manager.UseCase
  + UneFonction1UseCase.cs
    - Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

      Description générée automatiquement
    - ( Celle-ci va appeler et utiliser UneFonction1() de la classe IUneClasse.cs )
* Manager.UseCase.Contract
  + IUneFonction1UseCase.cs
    - Pareil que le contrat de UneClasse mais pour le UseCase
* Pages
  + Page.cshtml.cs
    - Variable private readonly IUneFonction1UseCase \_lafonction
    - Le constructeur va prendre comme paramètres :
      * IUneFonction1UseCase UneFonction1UseCase
    - Et il va changer :
      * \_lafonction = UneFonction1UseCase
    - Contient une fonction qui s’active lors d’un clique sur le bouton
      * \_lafonction. UneFonction1(string Chaine)

FINALEMENT, lorsque sur la page on clique sur un bouton pour utiliser une méthode de la classe UneClasse.cs, ce qu’il se passe c’est :

1. Clique sur le bouton
2. Appel d’une fonction de la page.cshtml.cs
3. Cette fonction va utiliser la méthode de l’interface UseCaseContract : \_lafonction. UneFonction1(string Chaine)
4. Le UseCaseContract de type IUneFonction1UseCase va faire référence à la méthode du UseCase UneFonction1UseCase.cs
5. Le UseCase va faire référence à une méthode de l’interface IUneClasse.cs
6. IUneClasse.cs va faire référence à la fonction UneFonction1() de UneClasse.cs\*

* Réalisation d’une série de test pour tester les fonctions et/ou méthodes

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

Description générée automatiquement

**Jour 16 (8 juin 2023)**

* Déploiement de l’image docker, beaucoup de choses à changer, de chemins, de builds…

Une image contenant texte, Appareils électroniques, capture d’écran, logiciel

Description générée automatiquement

Désinstallation et réinstallation de dotnet

Dotnet restore

docker build -t tomyrulliat/conversiontool .

* Lorsque je démarre mon application web, une page blanche se lance
  + Test et recherche dans le but d’une résolution de ce bug
* Lorsque je démarre le build, message d’erreur
  + Problèmes de chemin résolu
  + Problèmes non résolus :
    - Skipping project "/src/BL.Sante.MS.Interop.Models/BL.Sante.MS.Interop.Models.csproj" because it was not found.
    - The referenced project '../BL.Sante.MS.Interop.Models/BL.Sante.MS.Interop.Models.csproj' does not exist
    - Lors du lancement du programme docker sur Visual Studio, une page se lance mais rien n’est présent dessus
    - error CS5001: Program does not contain a static 'Main' method suitable for an entry point [/src/Conversion\_Tool/Conversion\_Tool.csproj]

**Jour 17 (9 juin 2023)**

* Résolution de problème :
  + Pour les 2 premiers problèmes non résolus du dessus, il fallait simplement ajouter dans le dossier app/ du conteneur un dossier pour chaque bibliothèque de classe et y insérer son fichier .csproj à l’intérieur pour que le programme le trouve lors du build.
  + Pour le troisième problème :
    - Réparation de Visual Studio (Via l’installeur)
    - Mise à jour de Visual Studio (Via l’installeur)
  + Pour le dernier problème :
    - J’ai d’abord changé le program.cs pour rajouter un Static Main mais cela ne marchait pas
    - J’ai donc vérifié dans le dossier app/ et en effet, il ne s’était pas mis à jour (ce que j’ai donc fait)
    - Les dossiers mis à jour, j’ai relancer et toujours ce même problème
    - J’ai fini par réussir : Il faut simplement lancer le build dans la solution entière (sauf qu’il n’y a pas de fichier Dockerfile dedans, alors je l’ai rajouté et tout fonctionne !)
* Documentation :

1. Télécharger la Solution bl.sante.interop.microservices
2. Télécharger Docker Desktop pour windows « https://docs.docker.com/desktop/install/windows-install/ »
3. Ouvrir l’invite de commandes en mode administrateur
4. Se positionner dans le répertoire bl.sante.interop.microservices (via la commande « cd NomDuDossier» jusqu’au répertoire de la solution «  cd bl.sante.interop.microservices »)
5. Entrer : « docker pull tomyrulliat/outildeconversion »
6. Entrer : « docker build -t tomyrulliat/outildeconversion . » (Le point à la fin est important !!!)
7. Entrer : « docker run -d -p 8080:80 --name OutilDeConversion tomyrulliat/outildeconversion »
8. Aller sur le navigateur puis aller sur : « http://localhost:8080/ »

**Jour 18 (12 juin 2023)**

* Dès lors, j’utilisais docker via windows (interface Docker Desktop) le problème étant que son utilisation professionnelle est soumise à licence, et qu’il faut donc utiliser la version linux qui est accessible, ainsi, la documentation d’installation change puisqu’il faut intégrer l’installation Linux et du docker associé
* Documentation :

1. Obtenir la Solution bl.sante.interop.microservices
2. Ouvrir l’invite de commandes en mode administrateur
3. Vérifier si votre ordinateur dispose d'une installation de Linux. Si ce n'est pas le cas, vous pouvez installer Windows Subsystem for Linux (WSL)  
   (Commande pour installation : « wsl –install »)
   1. Pour plus d’informations : https://learn.microsoft.com/en-us/windows/wsl/install
4. Pour rentrer sur la partie Linux si vous n’y êtes pas encore, entrer :
   1. « bash » ou « wsl »
5. Installer docker en entrant :
   1. « sudo apt-get update && sudo apt-get install -y docker.io »
6. Se positionner dans le répertoire bl.sante.interop.microservices   
   (via la commande « cd NomDuDossier» jusqu’au répertoire de la solution «  cd bl.sante.interop.microservices »)
7. Entrer :
   1. « docker pull tomyrulliat/outildeconversion »
   2. « docker build -t tomyrulliat/outildeconversion . »   
      (Le point à la fin est important !!!)
   3. « docker run -d -p 8080:80 --name OutilDeConversion tomyrulliat/outildeconversion »
8. Aller sur le navigateur puis aller sur : <http://localhost:8080/>

* Help :
  + See all containers
    - « docker ps »
  + See all docker images
    - « docker images »
  + Uninstall docker then install docker again
    - « sudo apt-get remove docker docker-engine docker.io »
  + Stop, start and restart the docker via terminal
    - « docker stop <CONTAINER ID> »
    - « docker start <CONTAINER ID> »
    - « docker restart <CONTAINER ID> »
  + Delete the docker and his image
    - « docker stop OutilDeConversion »
    - « docker rm OutilDeConversion »
    - « docker rmi tomyrulliat/outildeconversion »

**Jour 19 (13 juin 2023)**

* Gestion des noms de fichiers qui provoquaient une erreur à l’exécution du programme
* Discussion avec Philippe sur la mise en place de mon projet sur la forge GIT de l’entreprise, pour qu’elle soit accessible en interne avec l’accompagnement de la documentation
  + PUSH SUR LE GITLAB (photo ci-dessous)

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

Description générée automatiquement

**Jour 20 (14 juin 2023)**

* Changement du PDF anonyme pour qu’il puisse être de type PDF/A1 (simple changement de pdf en base 64 dans la méthode d’anonymisation
* Apprentissage et documentation en vue d’une mise en place d’une fonction de multi-langage sur mon application web

**Jour 21 (15 juin 2023)**

* Ajout de la fonctionnalité multi-langage (en ajoutant un menu déroulant pour choisir) :
  + J’ai tout d’abord créer des fichier .json contenant des variables et leurs correspondances (voir ci-dessous)

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

Description générée automatiquement

* + J’ai ensuite eu des soucis au niveau du changement de page (il fallait que je puisse garder la langue au changement de page. Je suis passer par le localStorage contenu dans la page pour faire garder une variable selectedLanguage au changement de page. Mon soucis à été la difficulté de récupérer cette variable depuis un fichier .cshtml.cs (Il était préconisé d’utiliser jsRuntime mais il fallait rajouter beaucoup de code et cela ramenait encore beaucoup de dépendances). En effet, je ne pouvais pas utiliser simplement un window.localStorage.getItem(« selectedLanguage »).
  + Je suis donc passé sur des cookies qui permettent la même chose mais avec lesquels il est plus simple de récupérer la langue. On en arrive donc à une page entièrement fonctionnelle dans plusieurs langages (français, anglais et espagnol)

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

Description générée automatiquement

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

Description générée automatiquement

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

Description générée automatiquement

* Ajout d’un manuel d’utilisation dans le site avec des captures pour faciliter la compréhension :

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

Description générée automatiquement

**Jour 22 (16 juin 2023)**

* Finalisation de la fonction multilingue (coté développement, code « plus propre », commentaires…)

**Jour 23 (19 juin 2023)**

* Ajout de la couleur/affichage du site pour correspondre aux attentes :  
  Berger-Levrault travaille sur plusieurs secteurs d’activités, dans leur charte graphique, chaque secteur est accompagné d’une couleur précise :  
  Une image contenant texte, capture d’écran, logiciel, Page web

  Description générée automatiquement
* La charte :  
  Une image contenant texte, capture d’écran, logiciel, nombre

  Description générée automatiquement
* Utilisation de css et d’un script + utilisation de cookies pour qu’au clic sur un des 3 onglets, celui-ci se met en gras quand on se trouve sur la page.
* Série de tests de l’utilisation de l’image docker sur une machine linux : problème, en local tout fonctionnait mais sur la machine, il ne trouve pas le fichier bootstrap associé au site

**Jour 24 (20 juin 2023)**

* Mon maître de stage et moi se sont rendu compte que la charte graphique avait changée, j’ai donc dû modifier ma page :  
  Une image contenant texte, capture d’écran, Police, conception

  Description générée automatiquement  
  Une image contenant texte, capture d’écran, logiciel, Page web

  Description générée automatiquement
* Ajout d’un logo de téléchargement :  
  

Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement

* Au niveau du code, il y avait plusieurs fois où le style était intégré directement dans le html, j’ai donc pour chaque objet, créé des classes pour pouvoir les styliser plus facilement (cela contribue aussi beaucoup à la propreté du code)

**Jour 25 (21 juin 2023)**

* Anonymisation dans l’exporteur IHEXDM (FAIT)
* Vérification minimale des CDA (fonction interne à l’entreprise Berge-Levrault)
* Au changement d’onglet, clear du fichier temporaire de l’utilisateur (FAIT)
* Changer uploader (FAIT)
  + Tic de langage (les utilisateurs ne sauront pas forcément ce que cela veut dire « fichier uploadé avec succès » je l’ai donc remplacé par « fichier récupéré avec succès »
* Utilisation URL (usage des singleton…)
  + Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

    Description générée automatiquement

**Jour 26 (22 juin 2023)**

* Finalisation du Clear du dossier temporaire au changement d’onglet
* Ajout du clear lors du changement de langage (très important puisqu’à chaque changement de langue, cela recréait un dossier temporaire ce qui surchargeait beaucoup l’application.
* Réglages sur l’anonymisation qui plantait lors de la mise en conversion d’un fichier IHEXDM.zip 🡪 Fichiers CDA différents avec d’autres lignes/balises à gérer qui sont maintenant pris en charge par le processus d’anonymisation
* Recherche sur la vérification minimale CDA
  + Problème d’url permettant la vérification, le programme se stop à la ligne 238 du fichier HttpHelper.cs
  + Pour régler le problème (à voir avec Phillipe)
    - Dans CDAR2Converter\_Response.cs ligne 52
      * public string size { get; set; } changé en :
      * public int size { get; set; }
    - Dans CDAHelper.cs ligne 268
      * if (hashAttendu.ToLower() != retour.hash.ToLower() || sizeAttendue != retour.size) changé en :
      * if (hashAttendu.ToLower() != retour.hash.ToLower() || sizeAttendue != retour.size.ToString())
    - Il y a eu un changement au niveau d’une variable SIZE qui était int et qui a été changée en string ce qui fait dysfonctionner mon programme

**Jour 27 (23 juin 2023)**

* Discussion avec Phillipe : réglage du problème de la vérification minimale CDA
* Vérification de tout le code.
* Aurevoir !

**Ce que je n’ai pas eu le temps de faire**

* Une série de test pour l’application
* Voir l’intégration continue via GitLab
* Effectuer une doc plus complète (+ un ReadMe.md précis)
* Effectuer une récupération de plusieurs logs
* Mettre en place la validation CDA de gazelle, voir liens :
  + <https://interop.esante.gouv.fr/gazelle-documentation/EVS-Client/user.html#accessing-validation-results-using-rest>
    - Il serait donc possible de récupérer le statut du CDA envoyé.