

Dossier Projet

Fil-Rouge Jalon 2

Application client-serveur

Windows Forms

API REST

{Consultation et gestion des actifs matériels du SI d'AMIO}

PATRICK NARDI

CDA07 | EPCF 2

9 février 2024

Projet de mise en situation professionnelle dans le cadre du
RNCP 31678 de niveau 6

Concepteur développeur d'application

ESRP AMIO 2isa Millau



L'association de formation professionnelle AM¹IO m'a chargé, à travers un cahier des charges reçu le 14 avril 2023, de simplifier la maintenance du parc matériel en améliorant la communication entre les utilisateurs et le Service Informatique. Cette requête incluait le développement de deux applications : une application web destinée aux utilisateurs et une application desktop exclusivement dédiée au Service Informatique. La première a été validée lors du premier jalon le 23 juin. Le présent rapport, correspondant à l'avenant du 4 octobre complétant ce deuxième jalon. Il se concentre sur l'application de bureau permettant la consultation et la gestion des actifs matériels à travers une API REST².

La requête formulée par l'association concernant l'administration du matériel informatique fait une distinction entre deux rôles au sein du Service Informatique, à savoir la consultation et la gestion, au sein des fonctionnalités proposées par l'application. Pour le premier rôle, il s'agit de la consultation avec la possibilité de filtrer les actifs par catégorie. Pour le second rôle, la gestion englobe la manipulation des entrées et sorties d'actifs, pouvant inclure ceux faisant l'objet d'un contrat de maintenance avec une entreprise externe.

Au début de cette phase initiale du second projet, j'ai procédé à l'identification des besoins du client en utilisant la méthode QQQQCP et à la détermination des fonctionnalités par le biais des cas d'utilisation de l'application. Ensuite, dans une seconde étape, j'ai élaboré des maquettes à l'aide de l'outil Figma, en m'inspirant de l'application GLP³ actuellement utilisée par le service informatique de l'association. L'objectif était de concevoir une interface utilisateur conviviale, similaire à celle de l'application web existante. Parallèlement à cela, j'ai rédigé les scripts de création de la base de données en SQL en vue d'un déploiement sur le système de gestion de base de données MariaDB, accompagné de son jeu de données représentatif.

Après avoir obtenu l'approbation du Product Owner pour la preuve de concept, j'ai élaboré le client desktop en utilisant le Framework .Net, avec une interface graphique basée sur Windows Forms. Pour accroître la flexibilité du client, j'ai opté pour une architecture MVC. En ce qui concerne l'API REST, j'ai suivi la conception en couches spécifiée dans le cahier des charges. Les deux architectures font appel à l'injection de dépendances, ce qui permet la réalisation de contrats d'interfaces, les rendant ainsi plus facilement maintenables et évolutives pour les futures demandes du client. Enfin, j'ai automatisé les tests en utilisant la bibliothèque xUnit, couvrant à la fois les tests unitaires et les tests d'intégration. Pour conclure la phase de développement, j'ai validé l'ensemble des fonctionnalités demandées par le biais de tests fonctionnels, réalisés avec la participation de différents utilisateurs représentatifs.

Actuellement, la première version du client est disponible, et l'API REST est également accessible en tant qu'image Docker sur mon DockerHub, en prévision de la phase d'acceptation par le service informatique de l'association AMIO.

Login

- *Consultation :*
- *Gestion :*
- *Sans droit :*

Mot de passe :

¹ <https://amio-millau.fr/>

² Interface de programme d'application <https://www.ibm.com/fr-fr/topics/rest-apis>

³ Logiciel de gestion de services open source <https://glpi-project.org/fr/>

Table des matières

1.	Liste des compétences du référentiel couvertes par le projet	6
2.	Méthode de travail	6
3.	Environnement technique de développement.....	6
4.	Phase analyse.....	7
4.1.	Analyse du besoin.....	7
4.2.	Analyse des flux de travail	7
4.3.	Planification	8
5.	Spécifications techniques et de sécurité	9
5.1.	Analyse des cas d'usage.....	9
5.2.	Scénarios des cas d'utilisation	12
5.3.	Spécifications de l'architecture logiciel	13
5.3.1.	Architecture du client Windows Forms	14
5.3.2.	Contrat entre le client et le serveur.....	15
5.3.1.	Architecture du serveur	16
5.3.1.	Échange et validation des données entre serveur et client.....	17
5.3.2.	Traitement des exceptions	17
5.4.	Spécifications de la persistance des données	17
5.5.	Spécifications et conception du client Windows Forms.....	18
5.5.1.	La vue connexion	19
5.5.2.	Vue consultation des actifs matériels.....	19
5.5.3.	Vue de gestion des actifs matériels	20
5.6.	Spécifications et conception de l'API	21
5.7.	Spécification de sécurité.....	22
5.8.	Phase de déploiement.....	23
6.	Développement de la solution.....	24
6.1.	Développement de la persistance des données.....	24
6.2.	Développement du client Windows Forms	26
6.2.1.	Frontend	26
6.2.2.	Backend.....	28
6.3.	Validation des Data Transfert Object de requête.....	31
6.4.	Développement de l'API.....	32
6.4.1.	Détail sur la réalisation du Token access Jwt.....	32

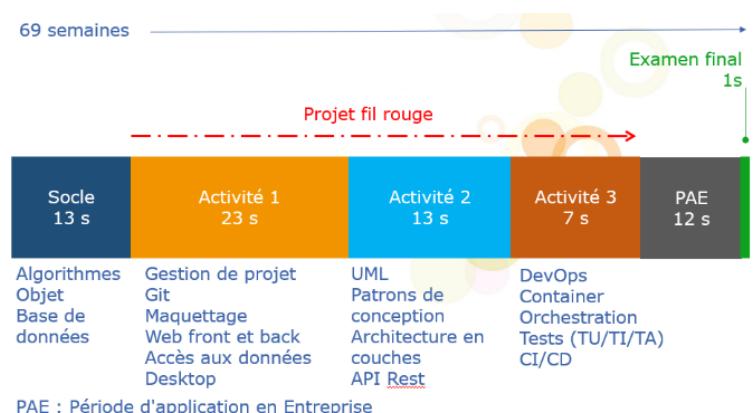
6.4.2.	Détail sur la réalisation d'une transaction.....	33
7.	Plan de tests et de déploiement	34
7.1.	Tests techniques	34
7.1.1.	Test unitaires	34
7.1.2.	Tests d'intégration	38
7.2.	Tests d'acceptation.....	39
7.2.1.	Tests fonctionnels.....	39
7.2.2.	Test de performance.....	41
7.3.	Déploiement	41
7.3.1.	Déploiement de l'API	42
8.	Veille, effectuée sur les vulnérabilités de sécurité	44
9.	Description d'une situation de travail ayant nécessité une recherche	46
10.	Conclusion.....	47
11.	Annexes.....	49
11.1.	Annexe 1, QOQCP	49
11.2.	Annexe 2, Phase d'ajout d'un actif matériel	53
11.3.	Annexe 3, passage de .NET 6 à 8	55
11.4.	Annexe 4, diagramme de séquence du flux d'ajout du client	56
11.1.	Annexe 5, Diagramme de classe pour la séquence d'ouverture de la vue de gestion.....	57
11.2.	Annexe 6, Exemple de test unitaire d'analyse du Token access Jwt.....	58
11.3.	Annexe 7, Métriques de la solution client-serveur	60

Le Fil-Rouge constitue la mise en situation professionnelle de notre groupe CDA07, dans le contexte de la formation professionnelle de Concepteur Développeur d'Applications (CDA). Au cours des quinze derniers mois passés au sein de l'ESRP 2isa, j'ai acquis des compétences professionnelles transférables en entreprise.

Ce projet desktop, débuté le mercredi 4 octobre 2023 et clôturé le vendredi 9 février 2024, m'a offert l'opportunité de mettre en pratique diverses compétences pertinentes dans le contexte professionnel :

- Le langage UML pour communiquer au travers de schéma,
- La conception logiciel avec deux architectures distinctes, MVC et n-tiers,
- Le développement d'une API REST avec le Framework .NET,
- L'emploi de la bibliothèque xUnit dans le cadre des tests unitaires et d'intégration,
- L'intégration en continue avec GitHub Action,
- Et le déploiement de l'API avec Docker desktop et le plugin Container.

Ces compétences fraîchement acquises seront cruciales pour la phase d'application en entreprise, qui débutera le 19 février et se poursuivra jusqu'au 11 mai. En collaboration avec les équipes de l'ESN Ntico à Montpellier, j'aurai l'opportunité de m'initier au métier d'Analyste d'exploitation, orienté vers le DevOps. En parallèle, je rejoindrai l'équipe de développement des solutions internes. Cette dernière phase constituera la conclusion de ma formation en vue de l'obtention du titre après la présentation orale lors de la semaine du 13 mai 2024.



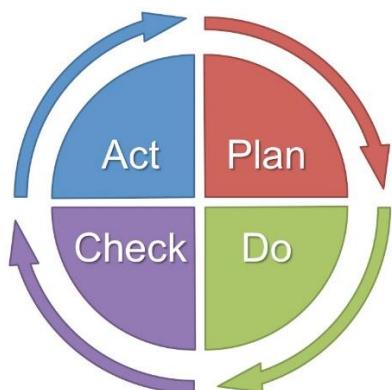
1. Liste des compétences du référentiel couvertes par le projet

Le concepteur développeur d'applications a pour responsabilité de concevoir et mettre en œuvre des services numériques en conformité avec les normes et standards de l'industrie. Il doit démontrer des compétences dans la conception d'interfaces utilisateur, le développement de bases de données, ainsi que dans l'architecture et le développement logiciel. Cela nécessite une capacité d'adaptation aux évolutions technologiques et une stricte observance des recommandations de sécurité.

La validation de ce titre requiert la présentation de livrables attestant de mon savoir-faire professionnel, correspondant aux blocs de compétences définis dans le Répertoire national des certifications professionnelles, sous la référence RNCP31678. Cela implique la démonstration spécifique des compétences pour le jalon actuel :

- Développer des composants métier,
- Contribuer à la gestion d'un projet informatique,
- Analyser les besoins et maquetter une application,
- Définir l'architecture logicielle d'une application,
- Concevoir et mettre en place une base de données relationnelle,
- Développer des composants d'accès aux données SQL et NoSQL,
- Préparer et exécuter les plans de tests d'une application,
- Préparer et documenter le déploiement d'une application

2. Méthode de travail



J'ai organisé les tâches à accomplir pour atteindre mes objectifs en suivant une approche similaire au cycle PDCA ou Roue de Deming (Planifier, Faire, Vérifier, Agir) tout au long des phases de conception et de développement du projet. Au cours de ces étapes, j'ai intégré de nouvelles connaissances acquises lors de formations, que j'ai ensuite appliquées à la solution présentée. Cette méthodologie m'a permis d'améliorer la qualité, respect des bonnes pratiques et la factorisation de mon code.

3. Environnement technique de développement

Liste des outils informatiques employés pour la réalisation du projet : IED : Visula Studio, Langage de développement : C# avec le Framework .Net 8, Base de données : MariaDB, Outil d'analyse ULM : Virtual Paradigm, Versionning : Git, Dockerisation : Docker Desktop et DockerHub.

4. Phase analyse

4.1. Analyse du besoin

J'ai appliqué la méthode QQQQCP pour analyser le cahier des charges et mettre en lumière les attentes du projet. Cette approche, symbolisant une technique d'analyse contextuelle, vise à comprendre une situation donnée en posant les questions appropriées (Qui, Quoi, Où, Quand, Comment, Pourquoi). Principalement utilisée lors de la phase de cadrage d'un projet, cette méthode m'a été bénéfique pour mieux appréhender la problématique spécifique, en me fournissant des informations précises et pertinentes.

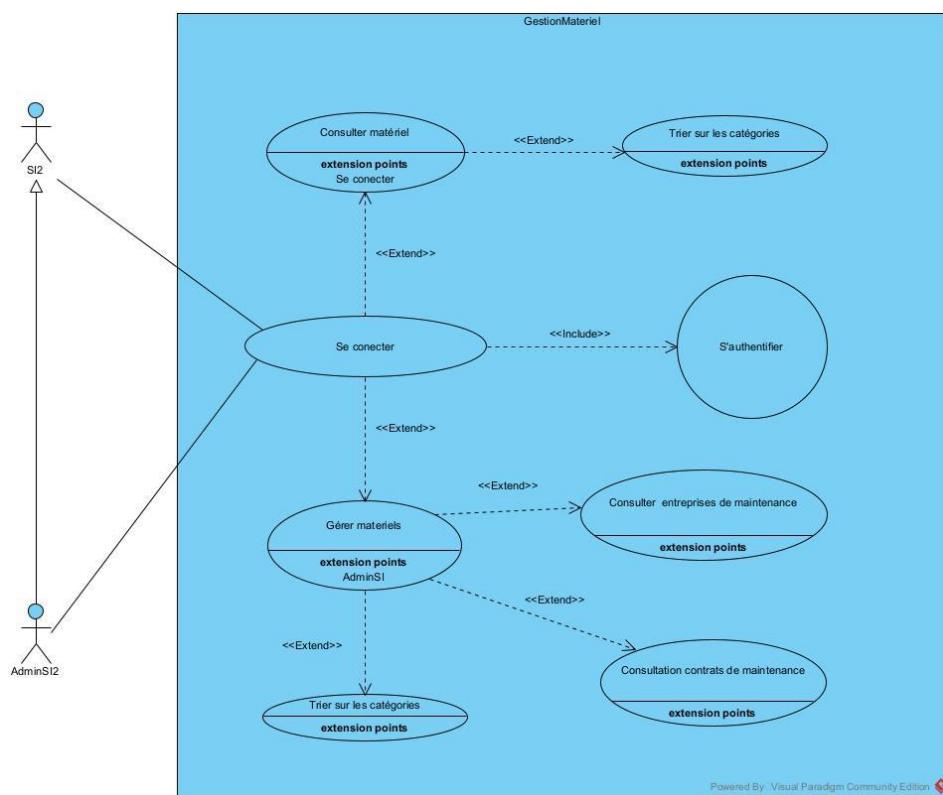
L'utilisation de cette méthode a permis d'adopter une approche réfléchie plutôt que de se précipiter dans la résolution du problème. En favorisant ma compréhension du projet, elle a également contribué à maintenir mon attention sur les objectifs à atteindre, comme documenté dans [l'annexe 1](#).

4.2. Analyse des flux de travail

Dans cette section, j'analyse les divers flux de travail exécutés par les utilisateurs de la solution. Pour ce faire, j'ai suivi une méthodologie orientée par les cas d'utilisation. Ma démarche s'est alignée sur les besoins des utilisateurs, depuis leurs spécifications jusqu'aux tests et à la livraison de la solution.

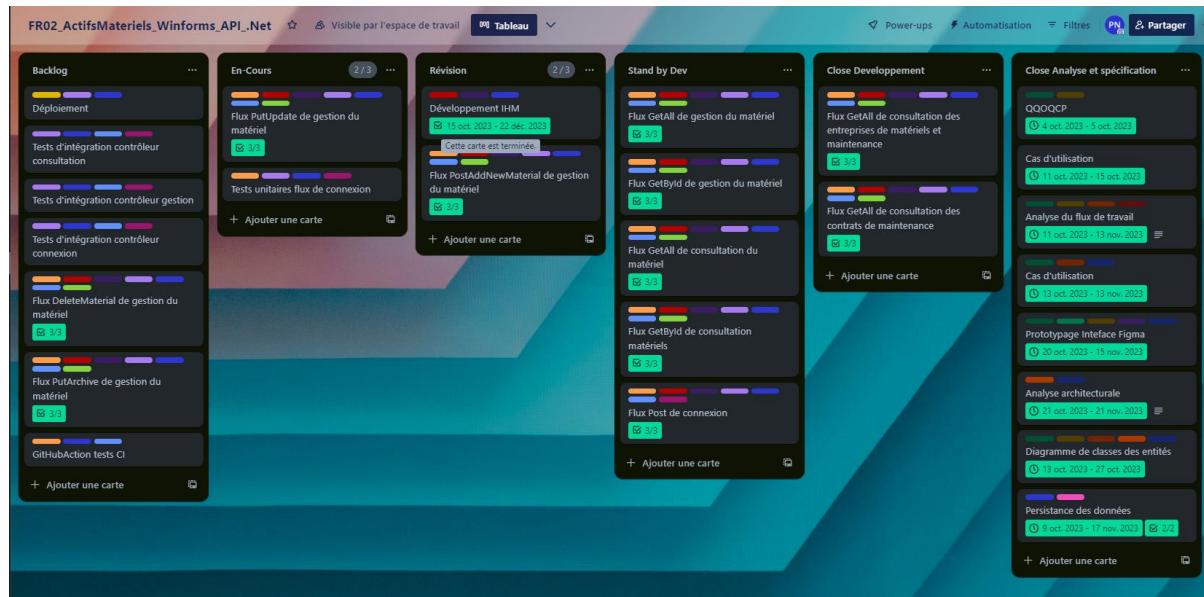
- Cas d'utilisation

Les cas d'utilisation reflètent les besoins fonctionnels du système informatique. Dans cette section, j'illustre, à l'aide du langage UML, les attentes fonctionnelles du système de gestion des actifs matériels destiné au service informatique de l'association AMIO.



4.3. Planification

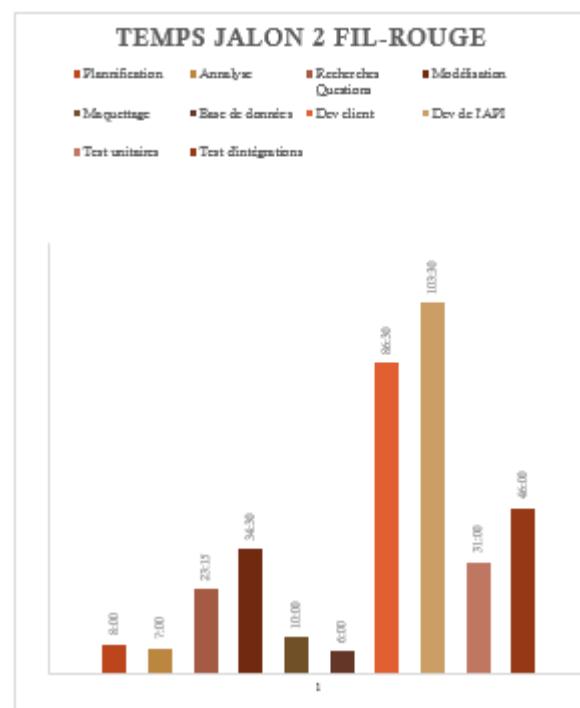
Pour structurer les diverses tâches et assurer une réponse efficace aux exigences du client, j'ai opté pour l'utilisation de l'application web Trello. Cette plateforme m'a permis de différencier les tâches à accomplir en vue d'atteindre mes objectifs. Tout au long du projet, j'ai adapté mon tableau de bord pour visualiser et anticiper mes réalisations. Voici mon planning début décembre :



J'ai également consigné quotidiennement mes heures de travail. Ce projet, lancé le 4 septembre 2023 et se clôturant le 9 février 2024, doit être accompli sur une période de 115 jours ouverts, parmi les 158 jours ouvrables. Au centième jour, le 11 janvier 2024, j'avais consacré 355 heures à ce projet.

Début le	04/09/2023
Fin le	09/02/2024
nb jours	158
nb jours ouvrables	115,00
nb h/j	
	actuel
	474
	355:45:00 heures
	3:33 heures/jour

Planification	8:00
Analyse	7:00
Recherches Questions	23:15
Modélisation	34:30
Maquillage	10:00
Base de données	6:00
Dev client	86:30
Dev de l'API	103:30
Test unitaires	31:00
Test d'intégrations	46:00



5. Spécifications techniques et de sécurité

5.1. Analyse des cas d'usage

Les User-Stories m'ont permis de décrire les besoins fonctionnels de l'application de consultation et gestion du matériel informatique de l'association AMIO, en adoptant le point de vue des utilisateurs finaux. Bien que je n'aie pas suivi la Méthode AGILE dans son ensemble, les User-Stories, en tant qu'artefact, représentent une solution pertinente pour un projet numérique.

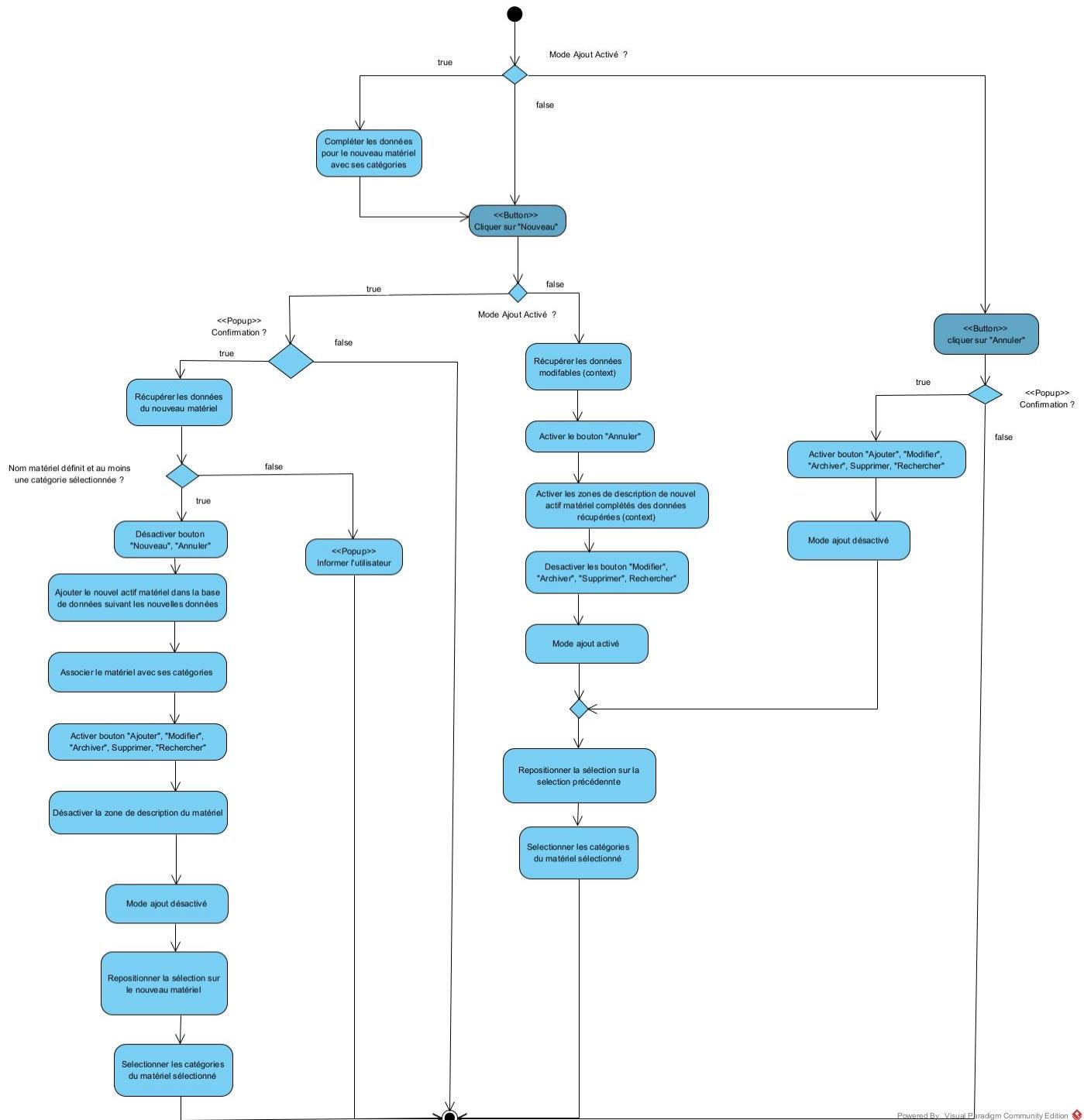
N°	En tant que	Je souhaite	Afin de	Comment
UC001	Personnel du service de maintenance du parc informatique de l'entreprise AMIO	Je souhaite pouvoir installer l'application sur mon poste de travail sous l'OS Windows 10 et supérieur.	Afin de réaliser le suivi et l'inventaire des actifs de matériel informatique	
UC101		Je souhaite pouvoir m'identifier à l'application Desktop de gestion des actifs de matériel informatique	Afin d'accéder à l'application des actifs matériels du service informatique de l'association AMIO	Par l'utilisation d'un login et un mot de passe via l'application desktop
UC102		Je souhaite pouvoir consulter la liste des actifs de matériel informatique	Afin de réaliser leurs suivis	
UC103		Je souhaite pouvoir trier la liste des matériels informatiques suivant les catégories matériel	D'obtenir uniquement l'affichage des actifs matériels de la catégorie sélectionnée	Par le choix d'une catégorie dans la liste des catégories
UC201	Administrateur du service de maintenance du parc informatique de l'entreprise AMIO	Je souhaite pouvoir ajouter du matériel informatique dans les actifs	Afin de compléter la liste des matériels informatiques disponibles	En définissant les caractéristiques nécessaires du matériel.

N°	En tant que	Je souhaite	Afin de	Comment
				Le nom doit-être définit obligatoirement avec au-moins une catégorie
UC202		Je souhaite pouvoir modifier les actifs de matériels informatiques présents dans la liste	Afin de compléter mettre à jour les matériels informatiques concernés dans la liste	En modifiant les caractéristiques matérielles actuelles. Avec une demande de confirmation
UC203		Je souhaite pouvoir archiver du matériel informatique	Afin d'archiver du matériel informatique disponibles de la liste	
UC204		Je souhaite pouvoir supprimer du matériel informatique	Afin de supprimer du matériel informatique disponibles de la liste	Avec une demande de confirmation
UC205		Consulter les contrats de maintenance avec des entreprises externes pour les actifs matériels sous contrat de maintenance	Afin de réaliser leurs suivis	
UC206 (Bonus)		Je souhaite pouvoir ajouter un contact de maintenance pour du matériel nécessitant une assistance externe		
UC207 (Bonus)		Je souhaite pouvoir mettre-à-jour un contact de maintenance pour du matériel nécessitant une assistance externe		

N°	En tant que	Je souhaite	Afin de	Comment
UC208 (Bonus)		Je souhaite pouvoir archiver un contact de maintenance pour du matériel nécessitant une assistance externe avec ses matériels informatiques accossés		
UC209		Consulter les entreprises de maintenance d'actifs matériel sous contrat de maintenance	Afin de réaliser leurs suivis	
UC210 (Bonus)		Je souhaite pouvoir ajouter une entreprise de maintenance de maintenance		
UC211 (Bonus)		Je souhaite pouvoir mettre-à-jour une entreprise de maintenance de maintenance		
UC212 (Bonus)		Je souhaite pouvoir archiver une entreprise de maintenance		
UC301	En tant que personnel d'AMIO	Je souhaite une IHM simple et intuitif	Afin de ne pas faire d'erreurs avec des informations non pertinentes	
UC401	En tant qu'association AMIO	Je souhaite une application sécurisée par authentification et autorisation.	Afin de donner accès aux utilisateurs authentifiés et autorisés	
UC402	En tant qu'association AMIO	Je souhaite une application facilement maintenable	Afin de faciliter les futures évolutions	

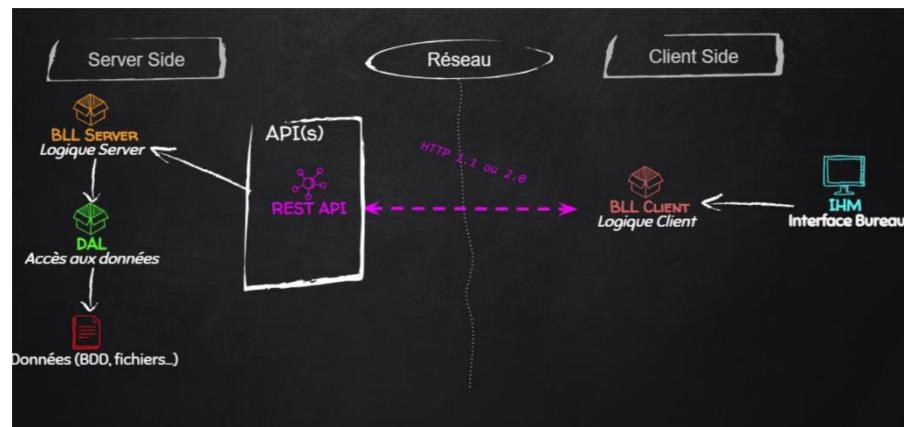
5.2.Scénarios des cas d'utilisation

Pour évaluer le déroulement des cas d'usage, j'ai utilisé le langage UML et son diagramme d'activité. Il m'a offert la possibilité de modéliser le cheminement des flux d'informations, de contrôle et de données au sein des scénarios des cas d'utilisations. Le diagramme d'activité du cas UC201, « *en tant qu'administrateur du service de maintenance du parc informatique de l'entreprise AMIO, je souhaite pouvoir ajouter du matériel informatique dans les actifs, en définissant les caractéristiques nécessaires du matériels* » analyse les relations et actions dans mise à jour de la vue lors d'un flux d'ajout de matériel synchrone.



5.3. Spécifications de l'architecture logiciel

Pour répondre aux exigences du client en matière de maintenabilité, j'ai conçu une architecture logicielle respectant la spécification de la page 8 du cahier des charges :



Outre cette architecture, j'ai délibérément adopté les principes [SOLID](#) pour créer une application flexible, maintenable, et plus aisément compréhensible par d'autres développeurs, dans le but de répondre à ses critères :

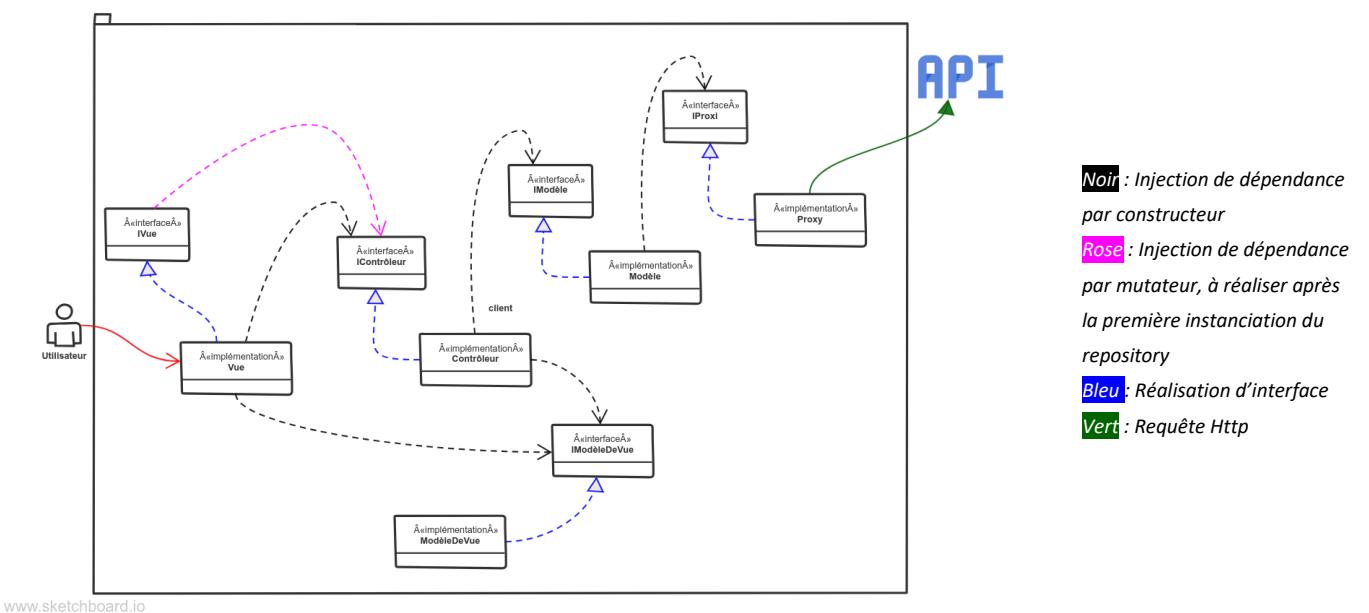
- S : les classes devront avoir une responsabilité unique (SRP) sans être d'une granulométrie trop petite. Pour ce faire, elles devront correspondre à un besoin spécifique. Pour exemple, l'encapsulation des données saisies par l'utilisateur ou encore celle des données pour la mise-à-jour de vue.
- O : pour l'OCP, le principe d'ouverture fermeture sera respecté par l'emploi d'interface avec seulement les signatures de méthode (ou fonctions), et des sous classes l'implémentant spécifiquement par le corps des méthodes.
- L : le LSP, principe de substitution de Liskov) sera lui respecté une hiérarchie de classe jusqu'aux propriétés abstraites. Ou chaque classe mère comportera uniquement les fonctionnalités communes à ses enfants. Au besoin elle peut être vide.
- I : le principe de ségrégation de l'interface, l'ISP devra être respecté par l'utilisation d'interface à responsabilité spécifique. Ainsi le couplage entre classe sera réduit et en relation avec le SRP, chaque interface devra avoir une responsabilité unique.
- D : Quand au dernier principe d'insertion de dépendance, le DIP sera respecté par l'utilisation du patron de conception d'inversion de dépendance. Elle sera réalisée via le constructeur. Cela réduira le couplage entre classes. Les implémentations dans les couches inférieures seront écrites de façon à dépendre des abstractions de plus haut niveau. En cas d'injection circulaire, voir avec le PO pour des injections par mutateurs.

5.3.1. Architecture du client Windows Forms

Par suite de discussions avec le Product Owner, j'ai orienté ma conception client vers le pattern de conception MVC en plus des spécifications mentionnées ci-dessus. Étant donné que Windows Forms n'est pas adapté à ce pattern, des ajustements dans la logique architecturale ont été nécessaires. Cette interface graphique, intégrée dans le Framework .NET, a exigé des adaptations pour son utilisation avec cette architecture :

1. La Vue (fenêtre issue de la classe Forms) réceptionne l'évènement
2. La Vue informe le Contrôleur qu'un évènement a eu lieu.
3. Le Contrôleur traite les données et demande au Modèle
4. Le Modèle consulte l'API au travers d'un Proxi
5. Le Modèle retourne les données issues de l'API au Contrôleur
6. Le Contrôleur traite les données et fabrique un Modèle de Vue
7. Le Contrôleur fournit à la vue un modèle de vue par lequel elle se met à jour

Le schéma ci-dessous intègre le principe architectural d'inversion de contrôle suivant la légende :



- La conception du client intégrera donc l'ensemble de ces spécifications citées précédemment, en plus de divers design pattern pour satisfaire aux mieux les principe SOLID.
- À cela, le flux d'information se fera en asynchrone, ainsi les composant resteront actif lors des temps de traitement avec une réduction du temps de traitement de séquence de requêtes lors des transactions. Les Vues auront par ce fait des états intermédiaires d'attente.
- Le client voyant l'API comme une boîte noire, ses requêtes seront en concordance avec le métier. Les règles métiers et la réponse à chaque cas d'utilisation se feront au sein de l'API.

5.3.2. Contrat entre le client et le serveur

Ce contrat représente les entrées de l'API que le client desktop devra appeler pour récupérer les données via des Data Transfert Object définis au [§5.3.1](#).

Les cas d'usage bonus ne sont pas pris-en-compte

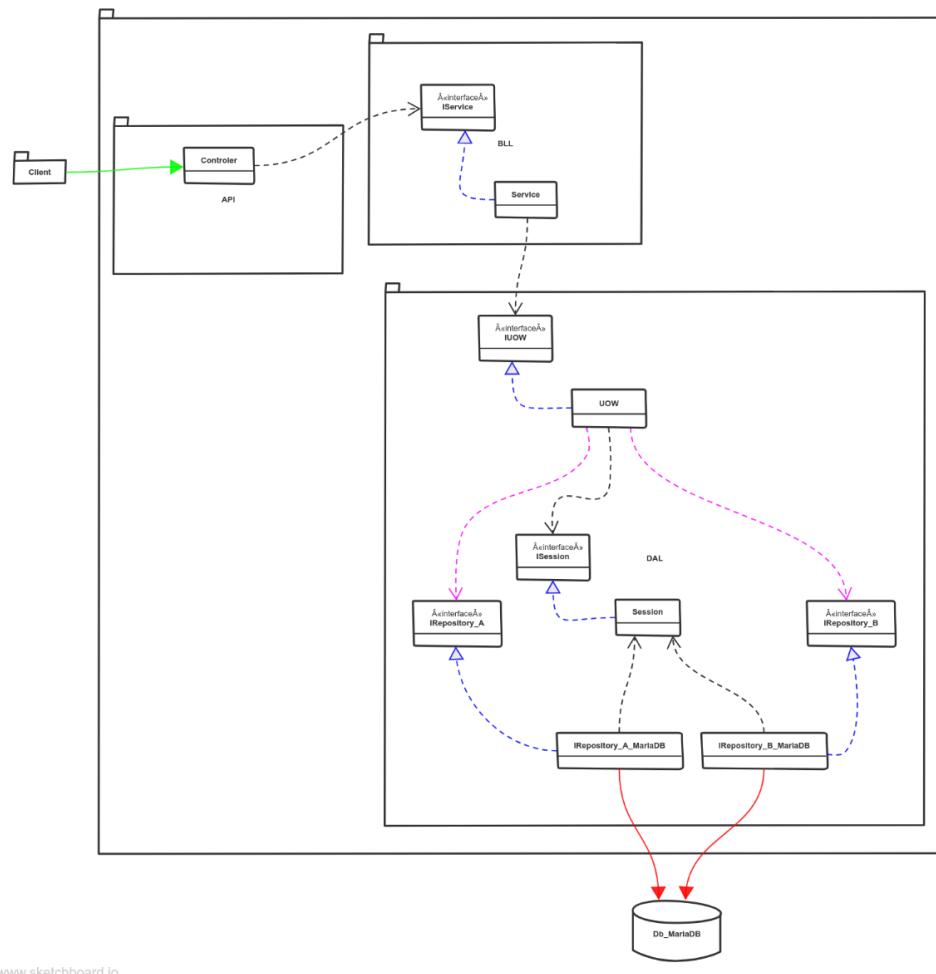
Rôle		Cas d'utilisation	Contrat suivant le protocole http 1.1 ou 2.0
Consultation	Gestion	UC101	End points
X	X	UC101	POST/api/Connexion
<i>Route de développement</i>		Route de développement	POST/api/Connexion/loginSwagger
<i>Route de développement</i>			GET/api/Connexion/testco
X	X	UC102	GET/api/Consultation
X	X	UC103	GET/api/Consultation/{idCategorie}
	X	UC205	GET/api/GestionCMaintenance
	X	UC209	GET/api/GestionEntreprise
	X	UC102	GET/api/GestionMateriel
		<i>Route de développement</i>	GET/api/GestionMateriel/materiel/{idMat}
	X	UC103	GET/api/GestionMateriel/categorie/{idCategorie}
	X	UC201	POST/api/GestionMateriel/add
	X	UC202	PUT/api/GestionMateriel/update
	X	UC203	PUT/api/GestionMateriel/archive
	X	UC204	DELETE/api/GestionMateriel/delete/{idmat}
		<i>Route de développement</i>	GET/api/GestionMateriel/deleted/{idmat}
		<i>Route de développement</i>	DELETE/api/GestionMateriel/delete/deleted/{idmat}

5.3.1. Architecture du serveur

Suivant les contraintes architecturales de l'API REST, récupérées sur des sites de confiance comme celui d'[IBM](#), et les conseils du Product Owner, j'ai conçu une architecture serveur en couches, respectant les principes suivants :

- Un contrat entre client et serveur,
- Des ressources échangées entre client et serveur (data transfert object, Dto) contenant l'ensemble des informations nécessaires au traitement sans être trop volumineuses,
- Un découplage client-serveur où chacun est indépendant de l'autre,
- L'application serveur n'est pas autorisée à stocker des données liées à une demande client,
- Une architecture en couches, API, BLL DAL avec l'emploi des patterns design Repository et Unit of Works pour traitement des transactions avec l'ORM Dapper,
- Architecture orientée service employant l'injection de dépendances.

Ci-dessous le schéma en UML du principe architectural de l'interface de programmation :



Noir : Injection de dépendance par constructeur

Rose : Injection de dépendance par mutateur, à réaliser après la première instanciation du repository

Bleu : Réalisation d'interface

Vert : Requête http

Rouge : MySql connexion

5.3.1. Échange et validation des données entre serveur et client

- Le serveur et le client Windows Forms, auront accès au projet des entités qui représente les données persistantes.
- Entre le serveur et le client, les données seront échangées par des objets spécifiques nommés Data Technic Object (Dto). Elles s'échangeront du proxy côté client au contrôleur côté serveur (point d'entrée de l'API) en étant serialisées en JSON.
- Au sein du client ou du serveur, les données seront échangées avec des objets de transfère (transians)
- La validation des données clients, sur les Dto de requête sera réalisée par la bibliothèque FluentValidation.AspNetCore V11.3.0.

5.3.2. Traitement des exceptions

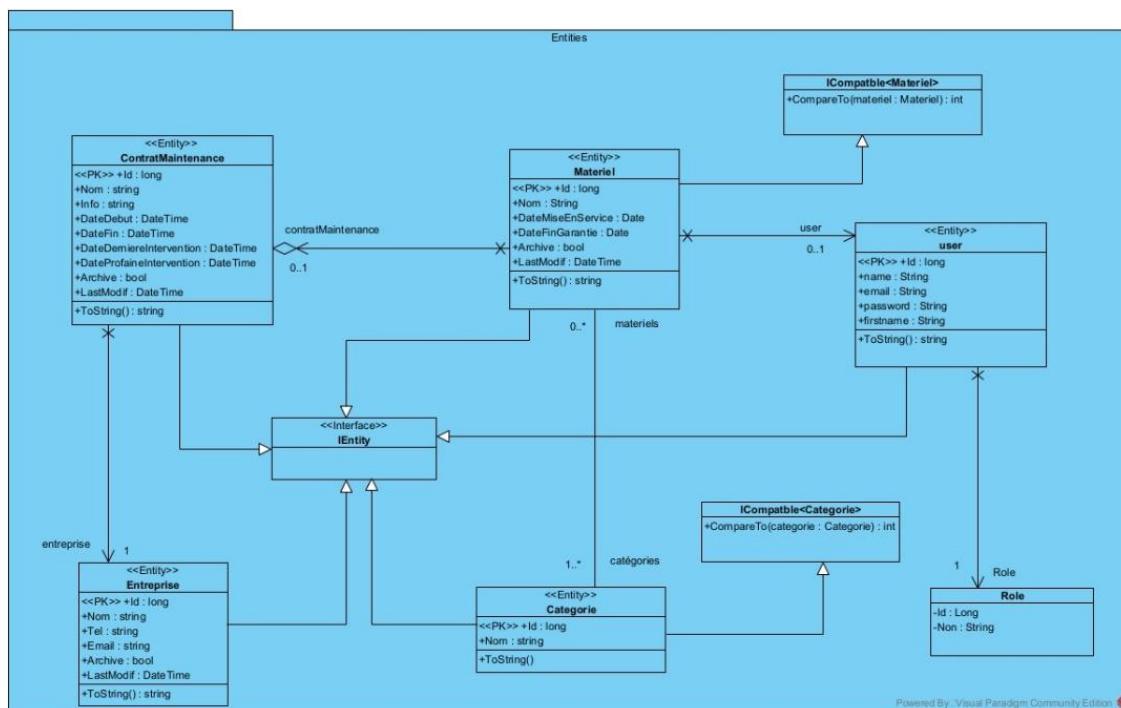
Le Product Owner demande l'utilisation d'un filtre d'exception dans l'API. employant classe [ExceptionFilterAttribute](#) de la bibliothèque Microsoft.AspNetCore.Mvc.Filters. Ainsi, l'ensembles des exceptions levées par le serveur seront sous-traités par ce filtre. De plus, elles seront reformatées, et leurs messages rendus plus explicites pour l'utilisateur du service informatique d'AMIO.

Par ailleurs, pour la vérification des données clients suivant les entités échangées entre l'API et le Client, j'ai employé la bibliothèque [Fluentvalidator](#). Celle-ci lèvera des exceptions dans le cas de non-correspondances dans les attendus des DTO de requêtes.

5.4. Spécifications de la persistance des données

Pour déterminer les données devants persister, j'ai d'abord réalisé un diagramme de classe permettant d'identifier les entités de la base de données. Ensuite, j'ai élaboré le script SQL de création de la base de données pour la solution desktop en lien avec la base de données réalisée en phase une pour l'application web.

- Pour ce faire j'ai réalisé le diagramme de classe des entités suivant :



- Et spécifier la persistance des données :

Attendus	Spécification de développement de la base de données
Écriture simple des scripts	Décomposition de la création de la base par étapes : <ol style="list-style-type: none"> 1. Suppression des tables si elles existent 2. Suppression des adaptations la table USERS 3. Création de la base de données et les clés primaire et secondaires 4. Adaptation la table USERS 5. Intégration du jeu de données 6. Script de création des vues
Intégration de la complexité est faite progressivement sur les liaisons entre tables	Respect des noms de tables, des noms des colonnes, de leurs types, des liaisons entre tables via les clefs étrangères Ajout de tables de liaisons et de flagues en fonctions des liaisons.
Respecte strict du diagramme de classe et transcription des liaisons entre tables.	Rédaction d'un jeu de données cohérent
Le jeu de données respecte les contraintes entre tables	

Attendus	Spécification de développement de l'API
Utilisation de l'ORM Dapper	https://github.com/DapperLib/Dapper

Pour la séparation des rôles demandée dans le cahier des charges par le client, elle se fera par les notions de consultation et gestion. La consultation aura des droits en lecture et la gestion ceux en lecture et écriture.

5.5. Spécifications et conception du client Windows Forms

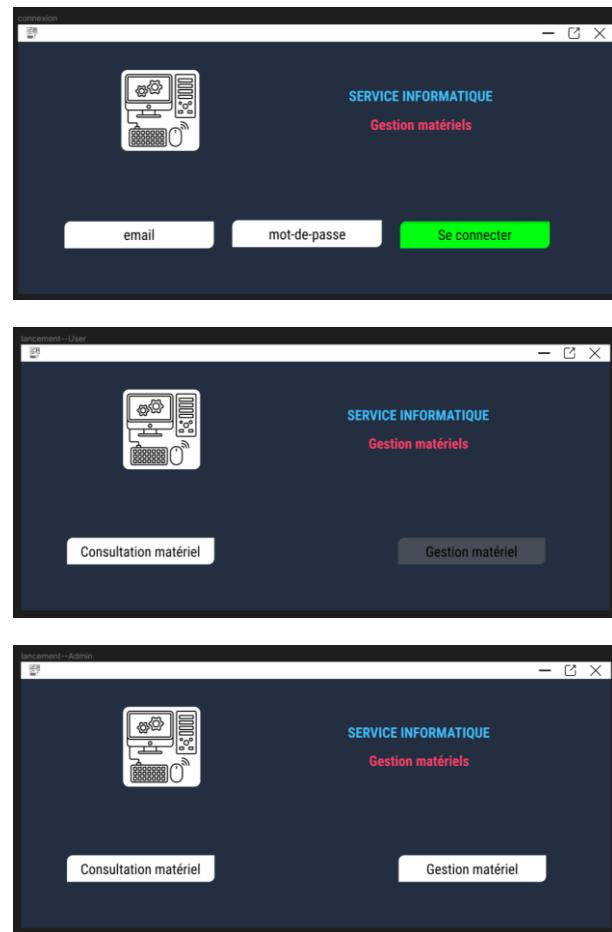
Le client doit reprendre la charte graphique du site web du jalon un. Il doit permettre aux différents types d'utilisateur que réaliser les cas d'utilisation qui leurs sont attribués. Pour répondre à ces besoins clients, j'ai fait le choix de différentier les responsabilités par les interfaces graphiques :

- La première permettant la connexion et l'ouverture des deux autres
- Une deuxième pour la consultation des actifs matériels
- Une troisième pour leurs gestions. Celle-ci n'étant accessible qu'aux administrateurs de gestion donnera accès aux actifs matériels, aux contrats de maintenance et aux entreprises associées.

5.5.1. La vue connexion

Le maquettage et réalisé avec l'outil web de design collaboratif Figma, accessible par ce [lien](#)

- Cette interface vise à autoriser le personnel du service informatique de l'association à se connecter au service des actifs matériels. Pour ce faire, ils doivent utiliser leur courriel professionnel ainsi que leur mot de passe. Une fois connecté, deux boutons seront disponibles en fonction de leurs rôles respectifs. Ils auront ainsi la possibilité d'accéder soit à la vue de consultation des actifs matériels, soit à celle dédiée à leur gestion.
- Cette vue répondra au cas d'usage UC101



5.5.2. Vue consultation des actifs matériels

- Pour cette vue, ses responsabilités sont la réalisation pur l'utilisateur des cases d'usage UC102 et UC103.
- Pour ce dernier, j'ai fait le choix dans un premier temps de multiplier le matériel suivant son nombre de catégorie. Ce choix fut revu lors de la réalisation de l'IHM.

Nom du matériel	Date de mise en service	Date de fin de garantie	Propriétaire actuel	Référence de l'enregistrement	Catégorie
écran	18/12/2022	18/12/2025	Jean Luc Marie	345	écran bureau
clavier	02/04/2023	02/04/2024	Claude Wallas	568	clavier cab
clavier	02/04/2023	02/04/2024	Claude Wallas	568	clavier cab
portable	04/07/2022	04/07/2025	Claudine de L'Anzai	1234	laptop
clavier	02/04/2023	02/04/2024	Claude Wallas	568	clavier cab
écran	18/12/2022	18/12/2025	Jean Luc Marie	345	écran bureau
écran	18/12/2022	18/12/2025	Jean Luc Marie	345	écran bureau
écran	18/12/2022	18/12/2025	Jean Luc Marie	345	écran bureau
écran	18/12/2022	18/12/2025	Jean Luc Marie	345	écran bureau

5.5.3. Vue de gestion des actifs matériels

Cette vue de gestion des actifs matériels comprend leurs contrats avec les entreprises fournisseurs de matériels informatiques. Ceux-ci les proposant sous contrat de maintenance.

Pour cette vue j'ai choisi d'employer une gestion avec onglets pour :

- Le premiers pour les matériels, et leurs catégories associées.

Cet onglet devant permettre pour un matériel :

- La consultation, UC102
- Le filtrage par catégorie, UC103
- L'ajout, UC201
- La modification, UC202
- L'archivage, UC203
- La suppression, UC205

- Le deuxième pour visualiser les contrats de maintenance, UC205

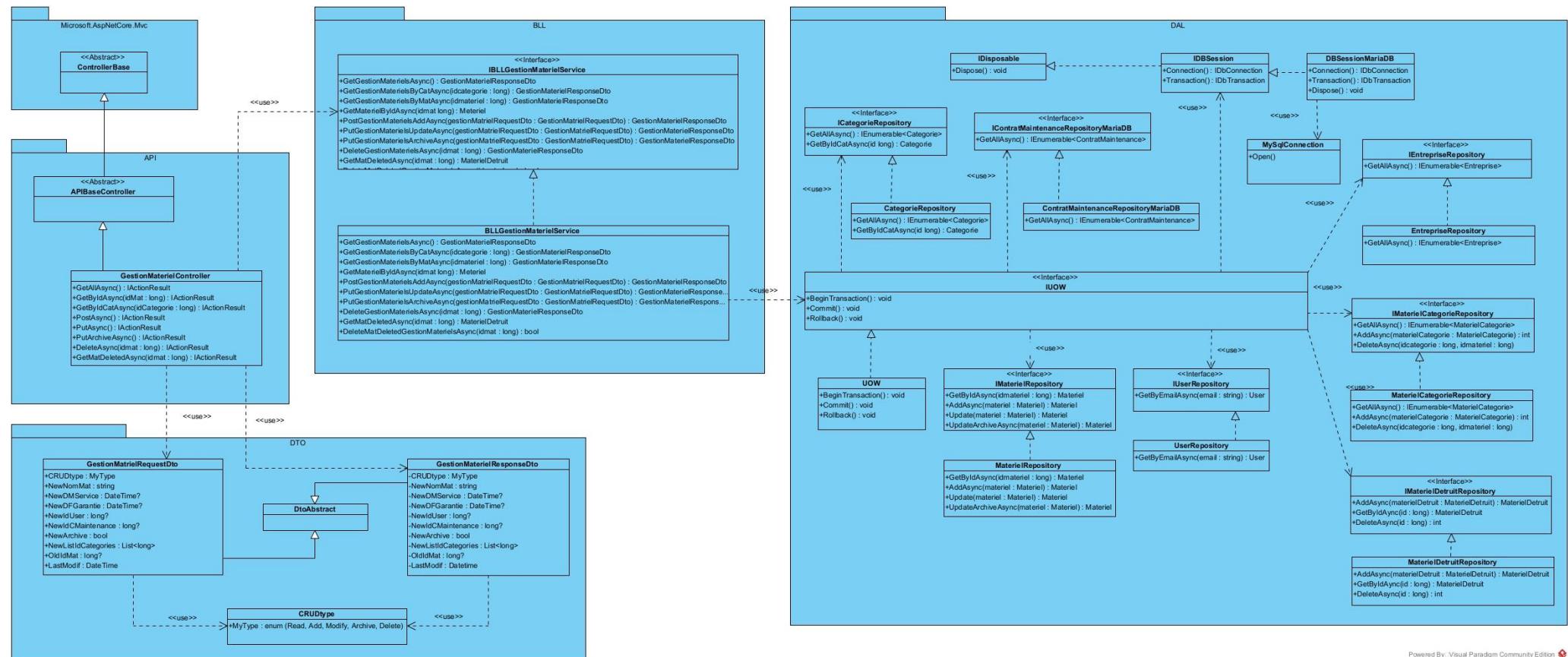
(Les cas bossus ne sont pas présentés)

- Le dernier pour visualiser les entreprises fournisseurs, UC209

(Les cas bossus ne sont pas présentés)

5.6.Spécifications et conception de l'API

Pour respecter l'architecture définie dans le cahier des charges, j'ai réalisé plusieurs diagrammes de classe avec le langage UML. Ils m'ont permis de spécifier les contrats d'interfaces dans l'architecture n-tier demandé par le client. Ci-dessous celui qui concerne les cas d'usage UC102, UC103, UC21, UC202, UC203, UC204, UC205 et UC209. Pour toutes les routes du contrat entre l'api et le client, du type : « /api/gestionmateriel/ ».



Powered By: Visual Paradigm Community Edition

5.7. Spécification de sécurité

Pour ce jalon, la demande concernant les besoins de sécurisation de l'application, or les bonnes pratiques de l'ANSSI⁴, sont définies comme tel par le Product Owner :

- L'emploi de la bibliothèque BCryt⁵,
- Des règles de visibilités des classes de chaque projet.
- De l'utilisation de Token access Jwt, avec la bibliothèque microsoft.aspnetcore.authentication.jwtbearer6 du Framework .Net.

Cela pour répondre au cas d'usage UC401.

- Détail sur le Token access Jwt (jeton web JSON) et sa spécification

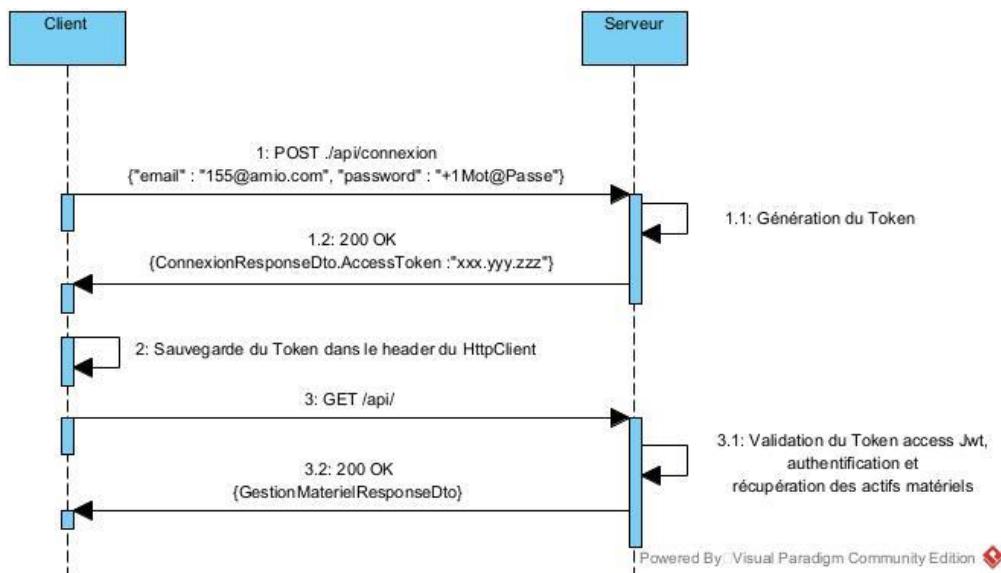
Le JWT est une méthode compacte et autonome de transmission sécurisée d'informations entre notre client et notre API sous la forme d'un objet JSON.

Ce jeton se compose d'une structure en trois parties encodées en base 64-URL séparées par un point.

- L'en-tête (header) contient le type (JWT) et l'algorithme de hachage utilisé, dans notre cas SHA256.
- La charge utile (payload) contient les revendications, qui sont des déclarations sur l'utilisateur et d'autres données.

La signature est formée par le chiffrement de l'en-tête codé, de la charge utile codée et d'une phrase secrète contenue dans le fichier de configuration appsettings.json. Elle est utilisée pour vérifier l'intégrité de l'expéditeur et du message.

Diagramme de séquence d'échange du Token access Jwt



Ce jeton permet une authentification de l'utilisateur. Dans le cas contraire l'API devra renvoyer une exception 401. Et permet de gérer les autorisations d'accès aux routes de l'API. Dans le cas contraire, elle devra renvoyer une exception 403.

⁴ <https://cyber.gouv.fr/10-regles-dor-pour-la-conception-et-la-mise-en-oeuvre-de-services-numeriques>

⁵ <https://github.com/BcryptNet/bcrypt.net>

⁶ <https://learn.microsoft.com/en-us/aspnet/core/security/?view=aspnetcore-8.0>

Les autorisations seront gérées suivant la spécification des rôles de Consultation et de Gestion, et suivant la bibliothèque Microsoft.AspNetCore.Authorization⁷ :

- Pour le rôle Consultations ⇒ Autorisation « USER »
- Pour le rôle de Gestion ⇒ Autorisations « USER, ADMIN »
- Ils seront intégrés dans le Claims du Token avec l'identifiant de l'utilisateur authentifié en tant que Claim Registered « ClaimTypes.NameIdentifier ».

5.8. Phase de déploiement

- Un déploiement de l'API sous container déployable sous Docker Desktop.
- La base de données elle devra être déployée sous le serveur local de l'association.
- Quant à l'application bureau, un exécutable de type X64 est demandé.

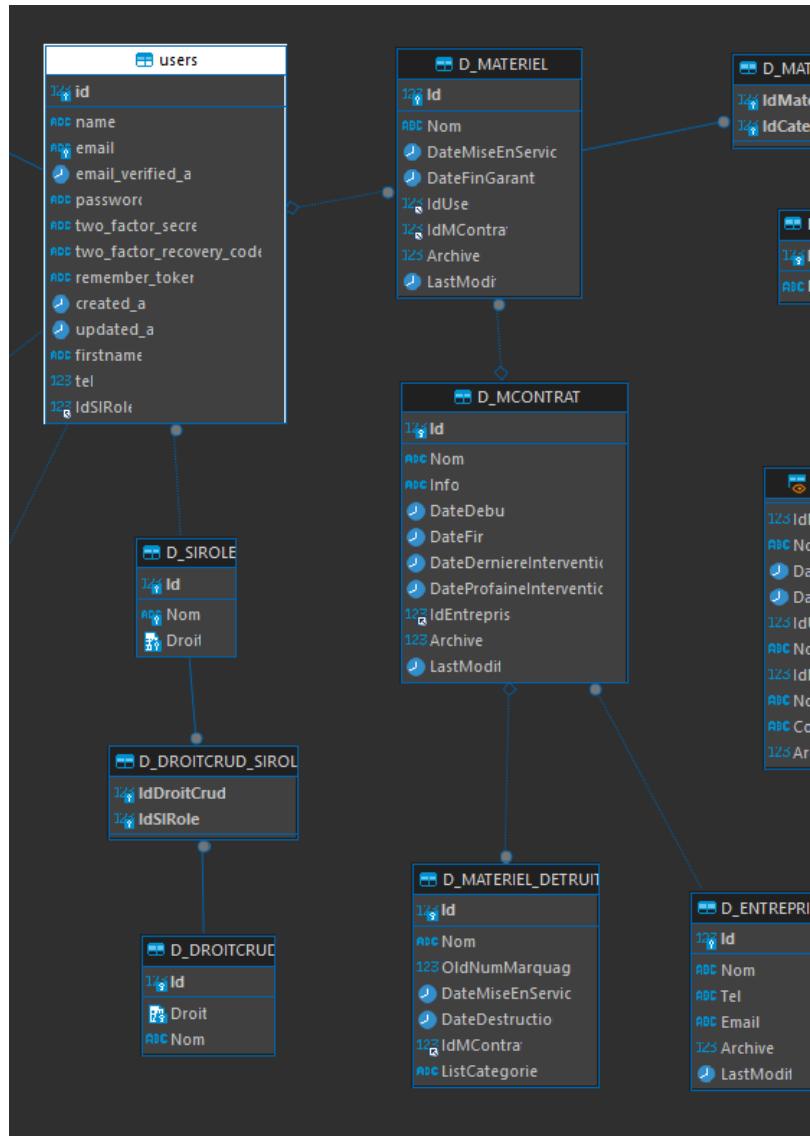
⁷ <https://www.nuget.org/packages/Microsoft.AspNetCore.Authorization/>

6. Développement de la solution

6.1. Développement de la persistance des données

Pour que la base de données puisse correspondre aux besoins du client, je l'ai conçu suivant les règles métiers fournis dans le cahier des charges et la spécification du [§5.4](#).

Ci-dessous les tables concernant l'application du jalon 2.



Dettes techniques :

- Nom de la base de données, GESTION_TICKETS. ➔ SIActifsMateriel ou SIAssetManagement
- Les table de liaison comme MatérielCatégorie, non pas d'identifiant unique. De ce fait les requêtes Dapper ne fonctionne pas correctement. Elles peuvent renvoyer « 0 » même si la requête c'est correctement exécuté :

```
var matCat = await
_db.Connection.QueryAsync<MaterielCategorie>(myQuery2, parameter, _db.Transaction);
```

Dans cette requête le résultat est « 0 » malgré la réussite.

Je l'ai remplacé par :

```
var matCat = await
_db.Connection.ExecuteAsync(myQuery2,
parameter, _db.Transaction);
```

- Création d'une table D_MATERIEL_DETRUIT et non une colonne dans la table D_MATERIEL de type booléen pour définir sa destruction.
- Création de tables de rôle pour la partie desktop sans lien avec ceux de l'application web.
- Emploie d'un DateTime en non d'un TimeStamp pour les requêtes optimistes

Exemple du script de création des tables et liste des fichiers pour la création des la base de données

```

desactivation des contraintes
SET foreign_key_checks = 0;
-- suppression de toutes les tables
DROP table IF EXISTS D_ENTREPRISE;
DROP table IF EXISTS D_MCONTRAT;
DROP table IF EXISTS D_MATERIEL;
DROP table IF EXISTS D_MATERIEL_DETRIUT;

DROP table IF EXISTS D_MATERIEL_CATEGORIE;
DROP table IF EXISTS D_CATEGORIE;

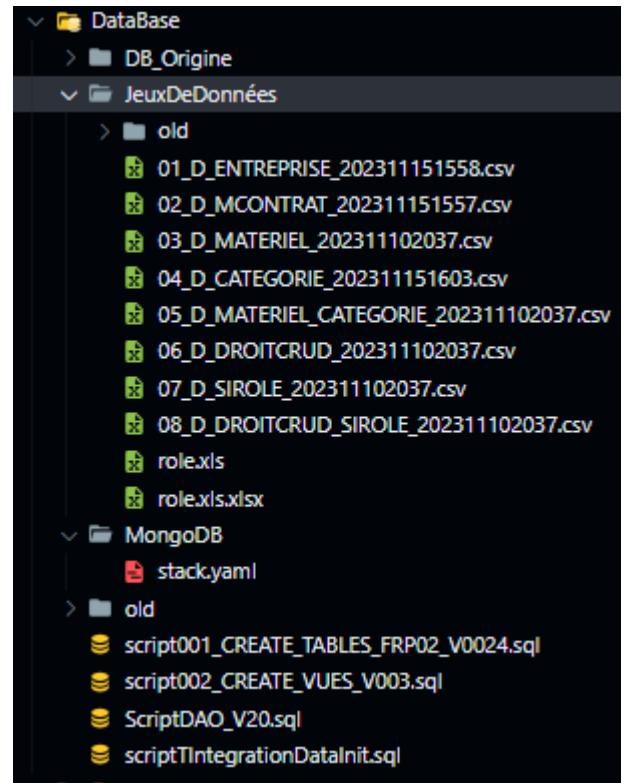
DROP table IF EXISTS D_DROITCRUD_SIROLE;
DROP table IF EXISTS D_DROITCRUD;
DROP table IF EXISTS D_SIROLE;

ALTER TABLE GESTION_TICKETS.users DROP FOREIGN KEY FK_SIRole_Id__Users;
ALTER TABLE GESTION_TICKETS.users DROP IdSIRole;

-- Création des tables
CREATE table D_ENTREPRISE (
    Id bigint(20) unsigned not null unique auto_increment,
    Nom varchar(50) not null CHECK (Nom <> ''),
    Tel? varchar(10),
    Email varchar(50) not null CHECK (Email <> ''),
    Archive boolean not null default false,
    PRIMARY KEY (Id)
);

CREATE table D_MCONTRAT (
    Id bigint(20) unsigned not null unique auto_increment,
    Nom varchar(50) not null CHECK (Nom <> ''),
    Info varchar(100),
    DateDebut Datetime not null default sysdate(),
    DateFin Datetime not null default sysdate() CHECK (DateFin >= DateDebut),
    DateDerniereIntervention Datetime CHECK (DateDerniereIntervention >= DateDebut),
    DateProfaineIntervention Datetime CHECK (DateProfaineIntervention >= DateDerniereIntervent
    IdEntreprise bigint(20) unsigned not null,
    Archive boolean not null default false,
    PRIMARY KEY (Id),
    FOREIGN KEY (IdEntreprise) REFERENCES D_ENTREPRISE(Id)
);
-- numéro de marquage = Id
CREATE table D_MATERIEL (
    Id bigint(20) unsigned not null unique auto_increment,
    Nom varchar(50) not null CHECK (Nom <> ''),
    DateMiseEnService? Datetime,
    DateFinGarantie? Datetime,
    IdUser? bigint(20) unsigned,
    IdMContrat? bigint(20) unsigned,
    Archive boolean not null default false,
    PRIMARY KEY (Id),
    FOREIGN KEY (idUser) REFERENCES users(id),
    FOREIGN KEY (IdMContrat) REFERENCES D_MCONTRAT(Id)
);

```



6.2.Développement du client Windows Forms

6.2.1. Frontend

Suivant mon prototypage avec l'outil web Figma, j'ai conçu les IHM correspondantes, avec les contraintes induites par l'utilisation des Forms du Framework .Net.

The screenshot displays three windows of a Windows Forms application:

- Vue connexion de l'application :** A login window titled "AMO SI - Actifs matériels". It features a logo, the text "SERVICE INFORMATIQUE Gestion matériels", and three buttons: "courriel", "*****", "Connexion" (highlighted in green), and "Fermer".
- Connexion →** An arrow indicating the transition from the login screen to the asset management interface.
- Vue Consultation des actifs matériel :** A list view titled "C Materiel" under "SERVICE INFORMATIQUE Consultation matériel". It shows a grid of assets with columns: Ref Marquage, Nom, Catégorie, Mise en service, Fin de garantie, Propriétaire, and Contrat. A dropdown menu "Catégorie" is open, showing categories like "chaise de bureau", "clavier", "CPU", etc. A "Filtrer" button is also present. At the bottom, there are buttons for "Nom", "Date mise en service", "Date fin garantie", "Propriétaire", "Ref marquage", "Contrat", and "Archive".
- Vue Gestion, actifs matériels**: A list view titled "Gestions" under "SERVICE INFORMATIQUE Gestion matériel". It shows a grid of assets with columns: Référence, Nom, Mise en service, Fin de garantie, Propriétaire, N° contrat de maintenance, and Nom contrat de maintenance. A dropdown menu "Catégorie" is open, showing categories like "laptop", "écran PC", "chaise de bureau", etc. A "Filtrer" button is also present. At the bottom, there are buttons for "Nouveau", "Modifier", "Rafraîchir", "Annuler", "Archiver", "Détruire", and "Fermer".

Vue Gestion, contrat de maintenance

The screenshot shows a Windows application window titled "Gestion" with the sub-section "SERVICE INFORMATIQUE Gestion matériel". Below the title, there are three tabs: "Matériel", "Contrat de maintenance", and "Entreprise de maintenance". The "Contrat de maintenance" tab is selected. A table lists six contracts:

Référence	Nom	Informations	nombre de matériels	Date de début	Date de fin	Dernière intervention	Prochaine intervention	Identifiant entreprise	Num entreprise	Archive
1	port_G07_2022	12 portables 17p... V...	0	01/01/2022	01/03/2025	11/10/2023	04/11/2024	1	LDLV	■
2	ecr_27p_2021	34 écrans 27p Vie...	1	04/02/2021	02/04/2025	04/02/2023	04/10/2024	2	materiel.web	□
3	cls_lur_G800	64 souris et clavier...	1	05/01/2022	05/01/2026	05/01/2023	05/01/2024	3	Dadasweb	□
4	mtpc_cpu_inter_Jc	32 cpu intel i9-G67...	0	05/02/2022	05/01/2026			4	Millauinfo	□
5	bur_chais_ergo	43 chaises ergo D...	1	05/04/2021	05/04/2026			5	idPc	□
6	ContratTest2	ContratTest2 Infos	2	03/01/2024 11:20	03/01/2024 11:20			5	idPc	□

At the bottom, there are several buttons: "Réinitialiser" (green), "Modifier" (green), "Rafraîchir" (blue), "Annuler" (red), "Actualiser" (green), "Définir" (green), and "Fermer" (red).

Vue Gestion, entreprises

The screenshot shows a Windows application window titled "Gestion" with the sub-section "SERVICE INFORMATIQUE Gestion matériel". Below the title, there are three tabs: "Matériel", "Contrat de maintenance", and "Entreprise de maintenance". The "Entreprise de maintenance" tab is selected. A table lists five companies:

Référence	Nom	Email	Tel	Nombre de contrats	Archive
1	LDLV	j.jean@ldlv.fr	534231265	1	■
2	materiel.web	info@materiel.web.fr	512986753	1	□
3	Dadasweb	allo@dadasweb.com	0564123765	1	□
4	Millauinfo	informatique@millau.info	0543228769	1	□
5	idPc	idpc@gmail.com	0543217654	2	□

At the bottom, there are several buttons: "Réinitialiser" (green), "Modifier" (green), "Rafraîchir" (blue), "Annuler" (red), "Actualiser" (green), "Définir" (green), and "Fermer" (red).

Vous trouverez en [annexe 2](#).l'exemple d'enchaînement des vues lors d'un ajout de matériel.

6.2.2. Backend

Pour analyser les séquences des différents flux, j'ai réalisé des diagrammes de séquence avec le langage UML. Vous trouverez en exemple dans l'[annexe 4](#), la séquence d'ajout d'un actif matériel.

Pour cette partie du développement, j'ai appliqué la logique suivante :

- Les contrôleurs portent la logique métier non gérable par la Business Logic Layer de l'api. Comme les demandes de confirmation ou la réception des exceptions afin d'afficher un popup d'information utilisateur.

De plus le client Windows Forms devait être développé avec une architecture de type :

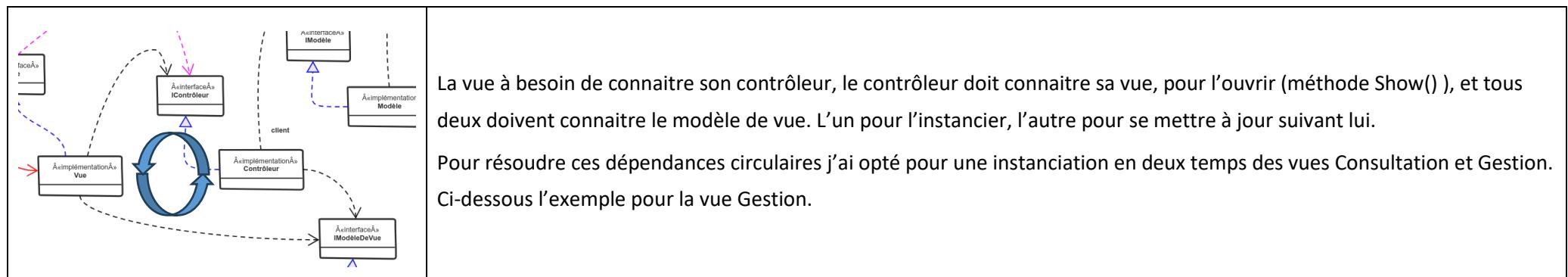
- Modèle Vue Contrôleur.

C'est deux principes ont induits :

- a) La réalisation d'interface par des classes concrètes et l'injection de dépendances
- b) Le traitement des cas d'erreurs dans les Contrôleurs du client.

Ces deux points avaient chacun de vraies opportunités d'apprentissages. Pour le premier, l'injections circulaires entre les vues, les contrôleurs et les modèles de vue. Pour le deuxième, la mise-à-jour des vues suivant une instance du modèle de vue.

- a) Injection circulaire (voir [annexe 5](#), diagramme de classe du flux d'ouverture de la vue gestion, représentatif de la solution pour l'injection circulaire) :



Dans le contrôleur, lors de l'ouverture de la vue, celui-ci s'injecte avec les contrôleurs des contrats de maintenance et des entreprises.

```
#region Ouverture de la vue
0 références
public async Task OpenViewGestion()
{
    if (!_viewGestion.GetIsOpen())
    {
        var responseTransian = await _mgestionMat.GetGestionMateriels();
        if (responseTransian is not null)
        {
            var myIMViewGestionMateriels = _mviewgestionmat.FactoIMViewGestionMateriels(null);

            myIMViewGestionMateriels.EnumMaterielsGestion = responseTransian.EnumMaterielsGestion;
            myIMViewGestionMateriels.EnumCategories = responseTransian.EnumCategories;
            myIMViewGestionMateriels.EnumMaterielCategories = responseTransian.IenummaterielCategories;
            myIMViewGestionMateriels.Exp = responseTransian.Exp;

            var viewGestionMat = _viewGestion.FactoViewGestion(myIMViewGestionMateriels, _messageBoxService);
            // Instanciation en 2 temps //\ 
            _viewGestion.ChangeIControllerGestionMateriel(this, _cgcm, _cge);

            _viewGestion.Show(viewGestionMat);
            // je définis la vue comme ouverte
            _viewGestion.ChangeValueOpen(true);
        }
    }
}
#endregion
```

```
/// <summary>
/// fonction d'implémentation post instantiation
/// </summary>
/// <param name="controllerGestionMetriel"></param>
/// <param name="controllerGestionMaintenance"></param>
/// <param name="controllerGestionEntreprise"></param>
2 références | Patrick Nardi, il y a 6 jours | 1 auteur, 1 modification
public void ChangeIControllerGestionMateriel(IControllerGestionMetriel controllerGestionMetriel, IControllerGestionCMaintenance controllerGestionCMaintenance,
| IControllerGestionEntreprise controllerGestionEntreprise)
{
    _cgm = controllerGestionMetriel;
    _cgcm = controllerGestionCMaintenance;
    _cge = controllerGestionEntreprise;
}
```

b) Le traitement des cas d'erreurs dans les Contrôleurs du client

Le client gère les exceptions et les cas d'erreurs de l'utilisateur dans l'application des règles métiers. Pour informer l'utilisateur avec un popup, j'ai développé un service de message `IMessageBoxService`, par lequel j'affiche l'ensembles des retours devant être affichés dans un popup.

```
///<summary>
/// Ajouter un matériel
///</summary>
///<param name="ValuesContext">Valeurs entrées par l'utilisateur</param>
///<returns>modèle de vue</returns>
///<exception cref="NotImplementedException"></exception>
1 référence
private async Task<IMViewGestionMateriel> PostNewMat(TrContextTabMateriel ValuesContext)
{
    // 1- je transforme le ContextTabMateriel en GestionMaterielRequestDto
    var gestionMaterielRequestTransian = TransformContext(ValuesContext);
    gestionMaterielRequestTransian.CRUDtype = MyType.Add;

    // 2- je demande confirmation à l'utilisateur
    var reponseQuestion = _messageBoxService.QuestionPopup(ValuesContext, Service.MessageBoxIcon.Type.Ajouter);

    if (reponseQuestion == DialogResult.Yes)
    {
        // 3- si oui j'appel le modèle avec le Dto et je récupère une response Dto
        var resultTransian = await _gestionMat.PostNewMat(gestionMaterielRequestTransian);
        if (resultTransian is null)
        {
            return MademViewGestionMateriel(null);
        }
        else if (resultTransian.ErrorMessage != null)
        {
            _messageBoxService.InformationErrorUserOK(resultTransian.ErrorMessage);
            return await BackupGestionMaterielWithContext(ValuesContext);
        }
        else
        {
            // 4- je transforme la réponse en modèle de vue
            var myView = MademViewGestionMateriel(resultTransian);

            // 5- je récupère le nouveau matériel par son id
            Materiel materielAdded = await GetMaterielById(resultTransian.IdMat);

            // 6- j'informe l'utilisateur de la réussite ou non de l'ajout
            var reponseInformation = _messageBoxService.InformationPopupMaterielOK(ValuesContext, materielAdded, resultTransian.CatOfIdMat, Service.MessageBoxIcon.Type.Ajouter);

            // 7- je retourne le modèle de vue avec le nouveau matériel sélectionné si réussite
            return myView;
        }
    }
    else return await BackupGestionMaterielWithContext(ValuesContext);
}
#endregion
```

```
2 références | markhor, il y a 7 jours | 1 auteur, 1 modification
public async Task<IMViewGestionMateriel> Ajouter(TrContextTabMateriel ValuesContext)
{
    try
    {
        if (!_viewGestion.GetModeAjoutIsActive())
            //if (!_viewgestionmat.SequenceAjouterIsActive)
            return await ActiverSequenceNewMat(ValuesContext?.IdMat ?? -1);
        else
            return await PostNewMat(ValuesContext);
    }
    catch (Exception e)
    {
        _messageBoxService.InformationErrorUserOK(e.Message);
        _viewgestionmat.Dispose();
        return _viewgestionmat.FactoMViewGestionMateriels(null);
    }
}
```

Ce service est implémenté par injection de dépendance dans le constructeur du contrôleur de gestion matériel.

```
0 références | markhor, il y a 7 jours | 1 auteur, 1 modification
public ControllerGestionMetriel(IModelGestionMateriels modelGestionMateriels,
    IMViewGestionMateriel mViewGestionMateriels, IViewGestion viewGestion, IMessageBoxService
    _messageBoxService, IControllerGestionCMaintenance controllerGestionCMaintenance,
    IControllerGestionEntreprise controllerGestionEntreprise)
{
    _gestionMat = modelGestionMateriels;
    _viewgestionmat = mViewGestionMateriels;
    _viewGestion = viewGestion;
    _messageBoxService = _messageBoxService;
    _cgc = controllerGestionCMaintenance;
    _cge = controllerGestionEntreprise;
}
```

6.3. Validation des Data Transfer Object de requête

Comme spécifié dans le [§5.3.1](#), les données des Dto reçus par l'API devront être validées suivant les exigences métiers. Comme la correspondance du courriel de connexion à un mail, ou encore que le nom d'un nouveau matériel ne pouvant pas être vide.

Pour ce faire j'ai réalisé des validations à que la bibliothèque FluentValidation⁸ réalise au début des méthodes des contrôleur de l'API.

Ajoutée en tant que service dans le builder de l'application, cette bibliothèque, au travers d'un dictionnaire, réaffecte les exceptions levées par l'API et celles retournées par la base de données.

La bibliothèque Microsoft.AspNetCore.Mvc, permet l'ajout de filtre sur des classes par les options du Builder.

```
[HttpPost()]
[ProducesResponseType(StatusCodes.Status201Created)]
[ProducesResponseType(StatusCodes.Status401Unauthorized)]
[AllowAnonymous] // Jst
3 références | markhor. il y a 3 jours | 1 auteur, 2 modifications
public async Task<ActionResult> LoginAsync([FromBody] ConnexionRequestDto connexionRequestDto)
{
    // 0- Le package validator vérifie le Dto et lève une exception si besoin
    ConnexionRequestDto requestDto = new(ConnexionRequestDto.Email, connexionRequestDto.Password);
    ConnexionRequestDtoValidator validator = new();
    var validationResult = validator.Validate(requestDto, options => options.ThrowOnFailures()); // 1 TestU Throw ValidationException

    // 1- j'appelle la BLL et sa fonction ConnexionUser en découpant le Dto dans les arguments
    // 2- Je crée le Dto de réponse
    // 3- je retourne le Dto de réponse serialisé dans l'ActionResult OK

    var trUser = await _connexionService.ConnexionUserAsync(connexionRequestDto.Email, connexionRequestDto.Password);

    if (trUser is null) return BadRequest(); // 2 TestU

    var reponse = new ConnexionResponseDto() // 3 TestU
    {
        Id = trUser.Id,
        NomPrenom = trUser.NameFormatted,
        Role = (Domain.Entities.Role.RoleEnum)trUser.Role,
        AccessToken = trUser.AccessToken,
        IsConnected = trUser.IsConnected,
        SettingType = trUser.SettingType,
    };

    return Ok(reponse); // Testunitaire
}
```

```
3 références | Patrick Nardi, il y a 2 jours | 2 auteurs, 3 modifications
public class ConnexionRequestDtoValidator : AbstractValidator<ConnexionRequestDto>
{
    //private Regex regexMailPattern = new Regex(patternMail);
    1 référence | Patrick Nardi, il y a 2 jours | 2 auteurs, 3 modifications
    public ConnexionRequestDtoValidator()
    {

        RuleFor(x => x.Email)
            .NotEmpty().WithMessage("votre courriel ne peut pas être vide")
            .EmailAddress().WithMessage("entrez une adresse de courriel valide")
            .NotNull().WithMessage("Ne peut pas être nul");
            //.Must(email => Email.Contains("@amio") == true);
        RuleFor(x => x.Password)
            .NotEmpty().WithMessage("le mot-de-passe ne peut pas être vide")
            .MinimumLength(5).WithMessage("Longueur minimum de 5 caractères")
            .NotNull().WithMessage("le mot-de-passe ne peut pas être nul");
    }
}
```

```
// Add services to the container.
builder.Services.AddControllers(options =>
{
    #if !DEBUG
    options.Filters.Add<ApiExceptionFilterAttribute>();
    #endif
});
```

⁸ <https://docs.fluentvalidation.net/en/latest/>

6.4.Développement de l'API

L'API, par son architecture, respect la spécification définie au [§5.3.1.](#)

6.4.1. Détail sur la réalisation du Token access Jwt

Côté Serveur

Les Tokens sont générés suivant les rôles de l'utilisateur authentifié, issus de la base de données.

```
string myToken = String.Empty;
if (myRole == Role.RoleEnum.Gestion)
    myToken = await Task.FromResult(GenerateJwtToken(user.id.ToString(), new List<string>() { "Admin", "User" })); // 5 TestU si role gestionnaire

else if (myRole == Role.RoleEnum.Consultation)
    myToken = await Task.FromResult(GenerateJwtToken(user.id.ToString(), new List<string>() { "User" })); // 6 TestU si role consultant

else
    myToken = await Task.FromResult(GenerateJwtToken(user.id.ToString(), new List<string>() { })); // 7 TestU si ni role gestionnaire ni consultant
```

Réalisation de la composition des Claims suivant spécification.

Instanciation du Token suivant le fichier de configuration appsettings.json

```
{
  "JwToken": {
    "JwtKey": "Henri_Mic",
    "JwtIssuer": "http://localhost:5001",
    "JwtExpireDays": 1
  },
}
```

Le Token sera renvoyé dans la ConnexionResponseDto par son attribut :

```
public string AccessToken { get; set; }
```

```
7 références
public virtual string GenerateJwtToken(string idUser, List<string> roles) // virtualisée pour être Mockée
{
    //var a = Parameters;
    //Add User Infos
    var claims = new List<Claim>()
    {
        new Claim(JwtRegisteredClaimNames.Sub, idUser), //ID User (Subject)
        new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString()), //ID Token
        new Claim(ClaimTypes.NameIdentifier, idUser) //ID User (NameIdentifier) == (Subject)
    };

    //Add Roles
    roles.ForEach(role =>
    {
        claims.Add(new Claim(ClaimTypes.Role, role));
    });

    var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["JwToken:JwtKey"]));
    var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

    //Expiration time
    //var expires = DateTime.Now.AddDays(Convert.ToDouble(myJwToken.JwtExpireDays));
    var expires = DateTime.Now.AddDays(Convert.ToDouble(_configuration["JwToken:JwtExpireDays"]));

    //Create JWT Token Object
    var token = new JwtSecurityToken(
        _configuration["JwToken:JwtIssuer"],
        _configuration["JwToken:JwtIssuer"],
        claims,
        expires: expires,
        signingCredentials: creds
    );
    //Serializes a JwtSecurityToken into a JWT in Compact Serialization Format.
    return new JwtSecurityTokenHandler().WriteToken(token);
}
```

Côté Client

Le Token réceptionné lors de la connexion, authentication, est sauvegardé dans la propriété static « AccessToken » du Proxi de connexion, puis il est ajouté au header du HttpClient.

```
// J'envoie la requête post à l'api comprenant l'objet de requête ConnexionRequestDto
HttpResponseMessage response = await _client.PostAsJsonAsync(_Uri, dtoRequest);
// je vérifie la réussit de la requête, si non lève une exception
response.EnsureSuccessStatusCode();
// je recupère et retourne le resultat en json en le transformant en ConnexionResponseDto
var reponse = await response.Content.ReadFromJsonAsync<ConnexionResponseDto>();
//enregistrement du token de connexion et inscription dans le header
AccessToken = reponse.AccessToken;
_client.DefaultRequestHeaders.Add("Authorization", "Bearer " + AccessToken);
```

6.4.2. Détail sur la réalisation d'une transaction

L'API se devait de supporter les règles métiers du service informatique de l'association AMIO.

Cette logique métier se trouve dans la Bisness Logique Layer, la bibliothèque de classe BLL dans le code.

Une de ces logiques est : un actif matériel doit être associé avec au-moins une catégorie.

Lors de la réalisation du cas d'usage UC201, l'ajout d'un nouveau matériel dans la table D_MATERIEL de la base de données, implique l'ajout de données dans la table D_MATERIEL_CATEGORIE pour associer le matériel avec des catégories existantes. Cette double action pour l'exécution de ce cas d'usage est réalisée dans une transaction dans le code source de la BLL par l'intermédiaire de l'Unit of Work, le _dbContext dans la méthode ci-contre.

Suivant l'analyse du §4.1, j'ai développé la fonction PostGestionMaterielsAddAsync(gestionMatrielRequestDto : GestionMatrielRequestDto) réalisant l'ajout de ces données au sein d'une transaction, et ainsi éviter les erreurs de persistance des données ou erreurs serveur, pour au besoin, revenir à l'état précédent.

Cette fonction fait appel à deux repositories, celui du matériel et celui des catégories.

Les deux classes concrètes exécutant les CRUD sur la base de données MariaDB pour les entités Matériel et Catégorie, réalisent les requêtes MySql avec le mapping object relationnel (ORM) Dapper.

Cette librairie disponible en package NuGet⁹, permet le mappage vers un objet ou un énumérable d'objet, de la réponse de la base de données.

Ci-contre, les fonctions d'ajout pour les repositories concernés dans le flux d'ajout matériel.

```
#region Post
    #references|l'muthor, il y a 2 jours | 1 auteur, 1 modification
    public async Task<GestionMaterielResponseDto> PostGestionMaterielsAddAsync(GestionMatrielRequestDto gestionMatrielRequestDto)
    {
        if (gestionMatrielRequestDto.CRUType != CRUType.MyType.Add)
        {
            throw new InsertEntityException(new Materiel());
        }
        if (gestionMatrielRequestDto.NewNomMat == string.Empty)
        {
            GestionMaterielResponseDto resultError = await Func GetAllGestionMateriels();
            resultError.ErrorMessage = "Vous devez nommer le matériel";
            resultError.IdMat = gestionMatrielRequestDto.OldIdMat ?? -1;
            return resultError;
        }
        else if (gestionMatrielRequestDto.NewListIdCategories.Count() < 1)
        {
            GestionMaterielResponseDto resultError = await Func GetAllGestionMateriels();
            resultError.ErrorMessage = "Le matériel doit avoir au moins une catégorie";
            resultError.IdMat = gestionMatrielRequestDto.OldIdMat ?? -1;
            return resultError;
        }
        else
        {
            // je transforme le Dto de requête en Matériel et liste de catégorie
            Materiel newMateriel = new()
            {
                Nom = gestionMatrielRequestDto.NewNomMat,
                DateMiseEnService = gestionMatrielRequestDto.NewDateMiseEnService,
                DateFinGarantie = gestionMatrielRequestDto.NewDateFinGarantie,
                IdUser = gestionMatrielRequestDto.NewIdUser,
                IdContrat = gestionMatrielRequestDto.NewIdContrat,
                //Archive = false,
                //LastModif = gestionMatrielRequestDto.LastModif,
            };
            // je commence la transaction
            _dbContext.BeginTransaction();
            // j'ajoute le matériel en récupérant l'id du nouveau matériel
            var newMat = await _dbContext.Materiel.AddAsync(newMateriel);
            // j'ajoute les catégories au matériel
            foreach (var idCat in gestionMatrielRequestDto.NewListIdCategories)
            {
                var newMatCat = new MaterielCategorie()
                {
                    IdCategorie = idCat,
                    IdMateriel = newMat.Id,
                };
                await _dbContext.MaterielCategorie.AddAsync(newMatCat);
            }
            // je ferme la transaction
            _dbContext.Commit();
            // je récupère les catégories du nouveau matériel si récupération de la liste listIdMatCat, inutile
            listIdMatCat = (await _dbContext.MaterielCategorie.GetAllAsync(newMat.Id)).ToList();
            // je fait la liste des catégories en fonction de la liste
            listCategorie listCat = new();
            foreach (var matCat in listIdMatCat)
            {
                var enumCats = await _dbContext.Categorie.GetByIdCatAsync(matCat.IdCategorie);
                listCat.Add(enumCats);
            }
            // je crée le résultat
            GestionMaterielResponseDto result = await Func GetAllGestionMateriels();
            result.IdMat = newMat.Id;
            result.CatIdMat = listCat;
            return result;
        }
    }
    #endregion
```

```
#region Post
    #references|l'muthor, il y a 9 jours | 1 auteur, 1 modification
    public async Task<Materiel> AddAsync(Materiel materiel)
    {
        string table = "D_MATERIEL";
        string items = "Nom, DateMiseEnService, DateFinGarantie, IdUser, IdContrat, Archive, LastModif";
        string variables = "@Nom, @DateMiseEnService, @DateFinGarantie, @IdUser, @IdContrat, @Archive, @LastModif";
        string myQuery = $"INSERT INTO {table}({items}) VALUES({variables}); SELECT LAST_INSERT_ID();";
        var parameters = new
        {
            materiel.Nom,
            materiel.DateMiseEnService,
            materiel.DateFinGarantie,
            materiel.IdUser,
            materiel.IdContrat,
            Archive = 0,
            LastModif = DateTime.Now,
        };
        var insertMat = await _db.Connection.QueryFirstOrDefaultAsync<long>(myQuery, parameters, _db.Transaction);
        if (insertMat <= 0) throw new InsertEntityException(materiel);
        materiel.Id = insertMat;
        return materiel;
    }
    #endregion
```

```
#region Post
    #references|l'muthor, il y a 9 jours | 1 auteur, 1 modification
    public async Task<MaterielCategorie> AddAsync(MaterielCategorie materielCategorie)
    {
        string myQuery2 = $"INSERT INTO D_MATERIEL_CATEGORIE(IdMateriel, IdCategorie) VALUES(@IdMateriel, @IdCategorie);";
        var parameter = new
        {
            materielCategorie.IdMateriel,
            materielCategorie.IdCategorie,
        };
        var matcat = await _db.Connection.ExecuteAsync(myQuery2, parameter, _db.Transaction);
        if (matcat <= 0) throw new InsertEntityException(materielCategorie);
        return matcat;
    }
    #endregion
```

⁹ <https://www.nuget.org/packages/Dapper/2.1.28/ReportAbuse>

7. Plan de tests et de déploiement

Remarque : À la suite du passage à la version 8 du Framework .Net, une régression dans les tests d'intégration est apparue (voir [annexe 3](#)). Cela est induit par un début de projet réalisé sur la version 6. De plus le GitHub Action ne passe plus, ne finit pas, pour les tests d'intégrations.

7.1. Tests techniques

J'ai appliqué au plus près, la méthodologie de test recommandée par Microsoft¹⁰, cela en employant le package xUnit Version 2.4.2¹¹. Ces tests techniques sont réalisés sur l'API, le client quant à lui sera testé lors des tests fonctionnels [§6.2](#). La base de données ne sera pas testée, hors attendus aux requêtes.

7.1.1. Test unitaires

Les tests unitaires sont un processus de vérification indépendante d'une unique unité de logiciel, elle-même définie comme étant la plus petite partie non divisible d'un code source. Dans le cas du flux de connexion cette logique a impliqué la réalisation de quatorze tests unitaires sur des parties atomique de mon code.

Ces tests ont induit des modifications de plusieurs types dans l'API :

- Architecturales, pour réduire la liaison avec la bibliothèque « Bcrypt »,
- De propriétés de projet, pour ne pas rendre une classe concrète « public »
- Des modifications des retours d'exceptions, pour correspondre aux standards du RFC 7231¹²
- Et de d'autres d'algorithmies,

J'ai réalisé ces modifications pour qu'elle puisse répondre à l'exigence de non-dépendances des classes entre elles, et permettre ainsi la réalisation de tests atomiques.

Test	Durée	Ca
▶ ✓ TestIntegrations (40)	1,6 min	
▶ ✓ TestUnitaireServeur (14)	471 ms	
▶ ✓ TestUnitaireServeur.TestController (3)	156 ms	
▶ ✓ ConnexionControllerUnitTest (3)	156 ms	
✓ Login_With_DTORequestIsNull_Should_Throw_ValidationException	145 ms	
✓ PostConnexion_Should_OKResponse	9 ms	
✓ PostConnexion_With_TransianUserIsNull_Should_BadRequest	2 ms	
▶ ✓ TestUnitaireServeur.TestService (11)	315 ms	
▶ ✓ BLLConnexionServiceUnitTest (11)	315 ms	
✓ ConnexionUser_With_NoRole_Should_UserTransitWithNoRoleInToken	2 ms	
✓ ConnexionUser_With_RoleConsultation_Should_UserTransitWithRoleUserInToken	2 ms	
✓ ConnexionUser_With_RoleGestion_Should_UserTransitWithRoleAdminAndUserInToken	104 ms	
▶ ✓ ConnexionUser_With_RoleInEnumRole_Should_UserTransitWithRoleInEnumRole (3)	24 ms	
✓ ConnexionUser_With_RoleInEnumRole_Should_UserTransitWithRoleInEnumRole(Rol...	2 ms	
✓ ConnexionUser_With_RoleInEnumRole_Should_UserTransitWithRoleInEnumRole(Rol...	2 ms	
✓ ConnexionUser_With_RoleInEnumRole_Should_UserTransitWithRoleInEnumRole(Rol...	20 ms	
▶ ✓ ConnexionUser_With_RoleNotInEnumRole_Should_UserTransitWithRoleNoRole (2)	172 ms	
✓ ConnexionUser_With_RoleNotInEnumRole_Should_UserTransitWithRoleNoRole(Rol...	2 ms	
✓ ConnexionUser_With_RoleNotInEnumRole_Should_UserTransitWithRoleNoRole(Rol...	170 ms	
✓ ConnexionUser_With_RoleNull_Should_UserTransitWithRoleNoRole	3 ms	
✓ ConnexionUser_With_SpecifiedMailPasswork_Should_UserTransitCorrectAttribut...	6 ms	
✓ ConnexionUser_With_VerifiedBCryptIsFalse_Should_Throw_UnauthorizeException	2 ms	

¹⁰ <https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-dotnet-test>

¹¹ <https://xunit.net/releases/v2/2.4.2>

¹² <https://www.rfc-editor.org/info/rfc7231>

Cas particulier du service de connexion, classe BLLConnexionService :

a. La réalisation des tests unitaires de

la BLL de connexion a impliqué
une modification architecturale.

Elle concerne la liaison de type

fort entre la classe

BLLConnexionService et la
bibliothèque Bcrypt¹³. Pour
distendre cette liaison, j'ai
implémenté une interface
IBCryptService avec comme
contrat la signature de la
méthode « Verify » de cette
bibliothèque.

Ci-contre les modifications
apportées à la fonction

ConnexionUser :

```

16  {
17 +     // injection de dépendance
18 +     private readonly IOW _dbContext;
19 +
20 +     public IConfiguration _configuration { get; }
21 +
22 -     public BLLConnexionService(IOW dbContext, IConfiguration configuration)
23 -     {
24 -         this._dbContext = dbContext;
25 -         this._configuration = configuration;
26 -     }
27 +
28     #region connexion
29 +
30 -    @<1,7 +44,7 @> public async Task<UserTransit> ConnexionUser(string email, string password)
31 -    {
32 -        // 2.1.2.2.1 je lève une exception personnelle
33 -        var user = await _dbContext.User.GetByEmailAsync(email);
34 -        bool verified = BCrypt.Net.BCrypt.Verify(password, user.password);
35 -
36 -        if (!verified) throw new UnauthorizedException(); // 1 TestU Fase =>
37 -        UnauthorizedException
38 -    }
39 +
40 +    public BLLConnexionService(IOW dbContext, IConfiguration configuration,
41 +                               IBCryptService bCryptService)
42 +    {
43 +        this._dbContext = dbContext;
44 +        this._configuration = configuration;
45 +        this._bCryptService = bCryptService;
46 +    }
47 +
48     #endregion
49
50     // aucune correspondance trouvée
51     if (!verified) throw new UnauthorizedException(); // 1 TestU Fase =>
52     UnauthorizedException
53
54 }
```

b. L'interface IConfiguration du package

Microsoft.Extensions.Configuration¹⁴ ne peut-
être Mocké. Par le fait du fichier
appsettings.json. Celui-ci comportant le
dictionnaire pour la réalisation du Token.

Pour ce faire dans un premier temps j'ai simulé
un dictionnaire identique à la configuration.

Puis après discussion avec le Product Owner j'ai
virtualisé la méthode qui réalise le token pour
pouvoir la moker.

```

3 références
public virtual string GenerateJwtToken(string idUser, List<string> roles) // virtualisée pour être Mockée dans les Tests
{
    /var a = Parameters;
    //Add User Info
    var claims = new List<Claim>();
    new Claim(JwtRegisteredClaimNames.Sub, idUser), //ID User (Subject)
    new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString()), //ID Token
    new Claim(ClaimTypes.NameIdentifier, idUser) //ID User (NameIdentifier) == (Subject)
};

//Add Roles
roles.ForEach(role =>
{
    claims.Add(new Claim(ClaimTypes.Role, role));
});

var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["JwtToken:JwtKey"]));
var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

//Expiration time
//var expires = DateTime.Now.AddDays(Convert.ToDouble(_configuration["JwtToken:JwtExpireDays"]));
var expires = DateTime.Now.AddDays(Convert.ToDouble(_configuration["JwtToken:JwtExpireDays"]));

//Create JWT Token Object
var token = new JwtSecurityToken(
    configuration["JwtToken:JwtIssuer"],
    configuration["JwtToken:JwtIssuer"],
    claims,
    expires: expires,
    signingCredentials: creds
);
//Serializes a JwtSecurityToken into a JWT in Compact Serialization Format.
return new JwtSecurityTokenHandler().WriteToken(token);
}
```

¹³ <https://github.com/BcryptNet/bcrypt.net>

¹⁴ <https://learn.microsoft.com/en-us/dotnet/api/microsoft.extensions.configuration?view=dotnet-plat-ext-8.0>

De plus je suis passé d'un « Fact » à une « Theory » pour correspondre à la logique métier de la BLL de connexion.

Le fait de virtualiser cette méthode de classe permet de ne pas la tester, comme dans l'état précédent de mon code (à gauche) :

```

50   ]
51   [Fact]
52   public async void ConnexionUser_With_RoleGestion_Should_UserTransitWithRoleConsu
53   {
54     // Arrange (arranger)
55     // Je simule un unit of work et la configuration
56     IUnitOfWork _unitOfWork = Mock.Of<IUnitOfWork>();
57     IConfiguration _configuration = Mock.Of<IConfiguration>();
58     IBcryptService _bcryptService = Mock.Of<IBcryptService>();
59
60     // Je simule le fichier de configuration appsettings.json par un dictionnaire correspondant à la configuration
61     var _memorySettings = new Dictionary<string, string>()
62     {
63       ["JwtKey", "Henri Michaux : Cette habitude, dis-je, je l'ai justement gardée."],
64       ["JwtIssuer", "https://localhost:50801"],
65       ["JwtExpirationDays", "1"]
66     };
67     IConfiguration _configuration = new ConfigurationBuilder().AddInMemoryCollection(_memorySettings);
68
69     // Je définit les valeurs d'entrée du test
70     var emailInputValue = "123@mein.com";
71     var passwordInputValue = "s3cr3tp@ssw3"; // Mot de passe
72     // Je définis le résultat attendu, utilisateur correspondant au couple email-mdp
73     var userExpected = new User()
74     {
75       id = 96106,
76       name = "Marc",
77       first_name = "Marc",
78       email = emailInputValue,
79       password = passwordInputValue,
80       idSIRole = 1
81     };
82
83     // Je simule
84     Mock.Setup(_unitOfWork).Setup(x => x.User.GetByEmailAsync(emailInputValue)).Returns(userExpected);
85     Mock.Get<IBcryptService>().Setup(x => x.Verify(passwordInputValue, userExpected));
86
87
88     // construction de l'instance à tester
89     var connexionService = new BLLconnexionService(_unitOfWork, _configuration, _bcrypt
90     );
91     var connexionServiceMock = Mock.Create<IBLLConnexionService>();
92     connexionServiceMock.Setup(x => x.ConnexionUser(emailInputValue, passwordInputValue));
93
94     // Act (réaliser)
95     var resultActual = await connexionService.ConnexionUser(emailInputValue, password
96
97     // Asset (vérifier)
98     Assert.Equal((Role)RoleEnum.userExpected.IdSIRole, resultActual.Role);
99   }

```

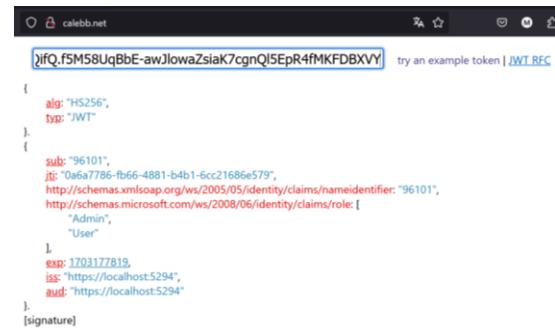
- c) La vérification de l'étape de génération du Token Jwt doit vérifier la bonne création des rôles utilisateurs.

Pour cela j'ai désérialisé celui-ci et vérifié la bonne attribution de des rôles.

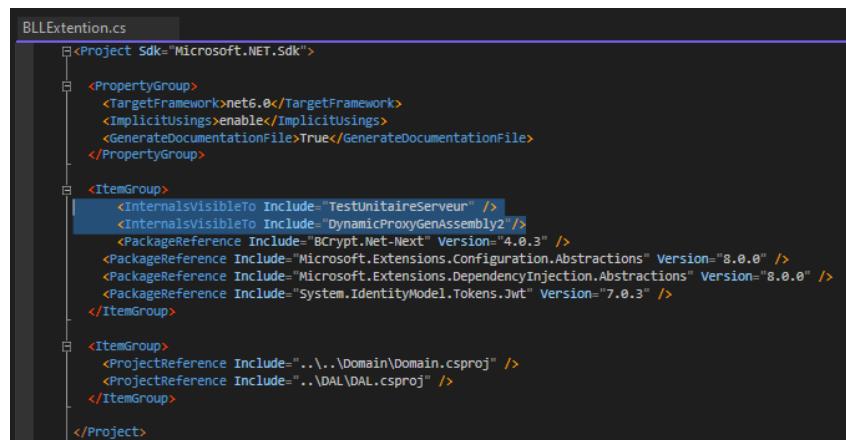
Ci-contre un exemple de Token fournit lors de la connexion.

Pour l'exemple, en [annexe 6](#) le test unitaire

ConnexionUser_With_RoleGestion_Should_UserTransitWithRoleAndUserInToken() qui analyse le Token access Jwt



- c) La gestion des autorisations est gérée par la réalisation de l'interface IBLLConnexionService, et réalise génération d'un Token d'authentification du porteur. La classe concrète de cette interface devant être mockée et ne pouvant être public, pour la réalisation des tests unitaires, j'ai ajouté un ItemGroup au projet afin que le projet de tests unitaires puisse y avoir accès sans diminuer la sécurité de l'application.



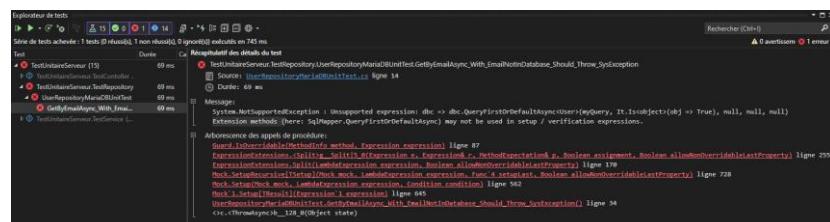
J'ai aussi ajouté dans ces ItemGroup, DynamicProxyGenAssembly2. C'est un assemblage temporaire construit par les systèmes de mocking qui utilisent CastleProxy comme Moq . Il est généré lorsque le simulacre est nécessaire, et éliminé une fois les tests terminés. Celui-ci me permet la réalisation des Moq de la génération du Token access

Cas particulier du repository User, classe UserRepositoryMariaDB

La DALL n'est généralement pas testée, du fait de sa très faible logique, et des multiples cas d'erreurs issues des requêtes vers le SGBD. Les erreurs vers ce dernier lèvent des exceptions supérieures à 500, gérées par le filtre d'exception. Malgré cela, mes tentatives de test de ce repository sont confrontées à une impossibilité technique de mocker des méthodes d'extension comme celle de l'ORM¹⁵ Dapper¹⁶. Dans le cas du flux de connexion, la méthode concernée est une de celle de la classe partiel SqlMapper :

```
/// <summary>
/// Execute a single-row query asynchronously using Task.
/// </summary>
/// <param name="cnn">The connection to query on.</param>
/// <param name="command">The command used to query on this connection.</param>
/// <remarks>Note: the row can be accessed via "dynamic", or by casting to an
/// IDictionary<string, object>.</remarks>
public static Task<dynamic> QueryFirstOrDefaultAsync(this IDbConnection cnn, CommandDefinition command) =>
    QueryRowAsync<dynamic?>(cnn, Row.FirstOrDefault, typeof(DapperRow), command);
```

Le mock de celle-ci renvoie une erreur informant le non-support des méthodes d'extensions :



Pour l'exemple, dans le cas de test des types de retours du SGBD indiquant l'absence du courriel entré comme login de connexion dans la base de données, j'ai dû modifier la méthode Task<User> GetByEmailAsync(string email) du repository User de mariaDB, pour qu'elle renvoie toujours un User. Et j'ai modifié la BLL pour qu'elle traite le cas du retour nul de la DALL.

```
public class UserRepositoryMariaDBUnitTests
{
    [Fact]
    public async void GetByEmailAsync_With_EmailNotInDatabase_Should_Throw_SysException()
    {
        // Arrange (arranger)
        // je simule la session
        IDBSession _db = Mock.Of<IDBSession>();
        // je simule la propriété connexion de IDBSession
        IDbConnection _cn = Mock.Of< IDbConnection>();
        Mock.Get(_db).Setup(db => db.Connection).Returns(_cn);

        // je définis l'email non présent dans la Database
        string email = "test@amio.com";

        // je simule la requête à réaliser
        var myQuery = @"SELECT Id, Name, Firstname, Email, Password, IdSIRole
                        FROM users
                        WHERE Email = @email";

        // je simule l'appel au SGBD MariaDB avec l'ORM Dapper
        Mock.Get(_cn)
            .Setup(dbc => dbc.QueryFirstOrDefaultAsync<User>(
                myQuery,
                //new { email },
                It.IsAny<object>(obj => true),
                null, null, null))
            .ReturnsAsync(new User());

        // je construit l'instance de classe
        var Repository = new UserRepositoryMariaDB(_db);

        // Act (réaliser)
        // je réalise et vérifie la levé de l'exception de type SysException
        await Assert.ThrowsAsync<SysException>(async () => await Repository.GetByEmailAsync(email));

        // Asset (vérifier) ci-dessus
    }
}
```

¹⁵ https://en.wikipedia.org/wiki/Object%E2%80%93relational_mapping

¹⁶ <https://github.com/DapperLib/Dapper>

7.1.2. Tests d'intégration

Les tests d'intégration ayant pour but la détection des erreurs non identifiées lors des tests unitaires, Ils doivent tester l'intégralité des erreurs pouvant apparaître dans un flux.

Dans le cadre de ces tests, seuls les flux concernant les contrôleurs de connexion, de consultation et de gestion des actifs matériel sont testés. Ceux concernant les contrats de maintenance, et les entreprises prestataires seront testées lors des tests d'acceptations.

```

public AbstractIntegrationTest(APIWebApplicationFactory fixture)
{
    /// Http client d'appel de l'API
    _client = fixture.CreateClient();
    _fixture = fixture;

    // je récupère le string de connexion dans le fichier appsettings.Integrations.json
    var connectionString = fixture.Configuration.GetSection("ConnectionString").Value;

    // j'instancie la connexion et je l'ouvre
    _db = new DBSessionMariaDBTest(connectionString);
    //Instanciation unique de la base de données par un bouble Lock
    if (count == 0)
    {
        lock (_locker)
        {
            if (count == 0)
            {
                DropCreateTablesInDatabase();
                DeleteAllElementsInDatabase();
                AddAllElementsInDatabase();
                count = 1;
            }
        }
    }
}

```

Chaque test devant se réaliser sans être dépendant d'un autre, tout en employant la même base de données de test ayant le même schéma qu'en production, avec un ordre d'exécution aléatoire, leur instanciation devait prendre en compte ces exigences.

Pour que mon code puisse y correspondre, j'ai initialisé la base de test au sein d'un double lock lors de l'instanciation de chaque test par leur classe mère abstraite `AbstractIntegrationTest`. Celle-ci créant une instance de `TestServer`¹⁷ pour chaque test, par injection de sa propriété de type `WebApplicationFactory`¹⁸, la fixture. Ainsi la base de données de test est déployée qu'une seul fois lors de la première instanciation de cette classe mère.

J'ai appliqué la même logique de lock pour la création des Token suivant les rôles. Cette création étant réalisé à chaque test, les lock évite la répétition de leurs créations.

Pour que ces tests couvrent une grande partie des cas d'utilisation normaux et dégradés, j'ai suivi un plan de tests des cas les plus probables pour chaque fonctionnalité de l'API testée. Par exemple, pour la l'ajout d'un matériel, UC201, j'ai réalisé les tests d'intégrations suivants :

Node normal	Mode dégradé
Rôle admin, gestion : <ul style="list-style-type: none"> • Valeurs correctes • Nom matériel vide • Matériel sans catégorie 	Rôle admin, gestion : <ul style="list-style-type: none"> • Id user inconnue • Id contrat de maintenance inconnue • CRUDtype non « add » Rôle user, consultation : <ul style="list-style-type: none"> • Valeurs correctes

¹⁷ <https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.testhost.testserver?view=aspnetcore-7.0>

¹⁸ <https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.testing.webapplicationfactory-1?view=aspnetcore-8.0>

7.2. Tests d'acceptation

7.2.1. Tests fonctionnels

Ces tests ayant pour objectif la validation des exigences et spécifications fonctionnelles, j'ai conçu une checklist vérifiant les sorties du client Windows Forms en fonction d'entrées utilisateur. Celles-ci réalisant les différents cas d'usage en mode normal et dégradé.

Ci-dessous les tests fonctionnels pour l'ajout d'une actif matériel, cas d'usage UC201 :

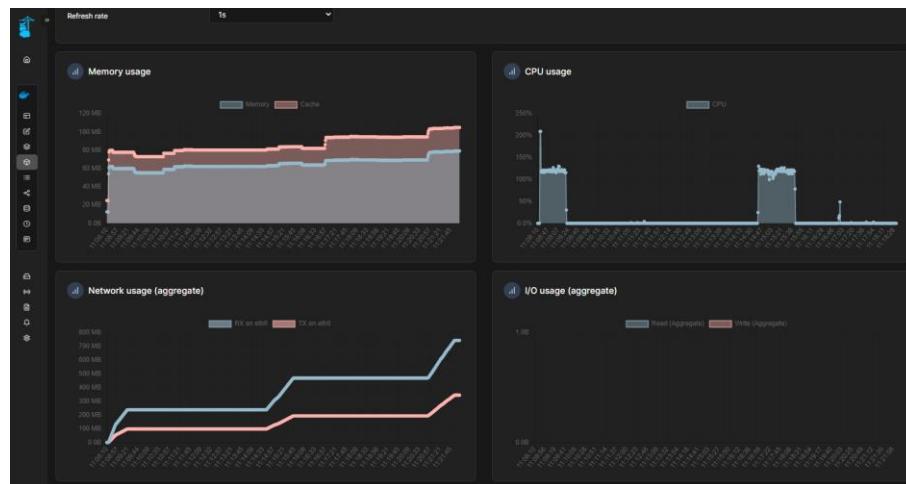
Cas d'usage	Données d'entrée :	Attendu	Validation
<p>UC201 :</p> <p><i>En tant qu'Administrateur du service de maintenance du parc informatique de l'entreprise AMIO</i></p> <p><i>Je souhaite pouvoir ajouter du matériel informatique dans les actifs,</i></p> <p><i>Afin de compléter la liste des matériels informatiques disponibles. En définissant les caractéristiques nécessaires du matériel.</i></p>	<p><i>Après connexion et clique sur le bouton « Gestion »</i></p> <p><i>Cliquez sur « Nouveau », puis complétez avec les données :</i></p> <p><i>Nom : « testAjout », Catégorie : « Laptop »</i></p> <p><i>Cliquez sur la demande de confirmation</i></p>	<p>Message de confirmation avec les données d'entrés</p> <p>Message d'information de l'ajout avec les données d'entrée et de sorties</p> <p>Repositionnement dans le tableau des actifs matériels sur le matériel nouvellement ajouté.</p> <p>Les boutons de la fenêtre redeviennent actifs sauf « annuler »</p>	✓
	<p><i>Après connexion et clique sur le bouton « Gestion »</i></p> <p><i>Cliquez sur « Nouveau », puis complétez avec les données :</i></p> <p><i>Nom : « testAjout », Catégorie : « Imprimante, scanner », Date de mise en service : «10/02/2023 », De de fin de garantie : nul, Propriétaire : T. Rodolf, Contrat de maintenance : nul</i></p> <p><i>Cliquez sur la demande de confirmation</i></p>	<p>Message de confirmation avec les données d'entrés</p> <p>Message d'information de l'ajout avec les données d'entrée et de sorties</p> <p>Repositionnement dans le tableau des actifs matériels sur le matériel nouvellement ajouté.</p> <p>Les boutons de la fenêtre redeviennent actifs sauf « annuler »</p>	✓

Cas d'usage	Données d'entrée :	Attendu	Validation
<i>Le nom doit-être défini obligatoirement avec au moins une catégorie</i>	<p><i>Après connexion et clique sur le bouton « Gestion »</i></p> <p><i>Cliquez sur « Nouveau », puis complétez avec les données :</i></p> <p style="padding-left: 40px;"><i>Nom : « testAjout », Catégorie : «»</i></p> <p><i>Cliquez sur la demande de confirmation</i></p>	<p>Message de confirmation avec les données d'entrés</p> <p>Message d'information indiquant que « Le matériel doit avoir au moins une catégorie »</p> <p>Repositionnement dans le tableau des actifs matériels sur le matériel nouvellement ajouté.</p> <p>Les boutons de la fenêtre redeviennent actifs sauf « annuler »</p>	✓
	<p><i>Après connexion et clique sur le bouton « Gestion »</i></p> <p><i>Cliquez sur « Nouveau », puis complétez avec les données :</i></p> <p style="padding-left: 40px;"><i>Nom : «», Catégorie : « Laptop »</i></p> <p><i>Cliquez sur la demande de confirmation</i></p>	<p>Message de confirmation avec les données d'entrés</p> <p>Message d'information indiquant que « Vous devez nommer le matériel »</p> <p>Repositionnement dans le tableau des actifs matériels sur le matériel nouvellement ajouté.</p> <p>Les boutons de la fenêtre redeviennent actifs sauf « annuler »</p>	✓
Mode dégradé	<p><i>API déconnecté, Client fenêtre Gestion Tab Matériel</i></p> <p><i>Clique sur « Nouveau »</i></p>	<p>Message d'information indiquant : « Aucune connexion n'a pu être établie car l'ordinateur cible l'a expressément refusée. (« adresse du serveur »)</p>	✓

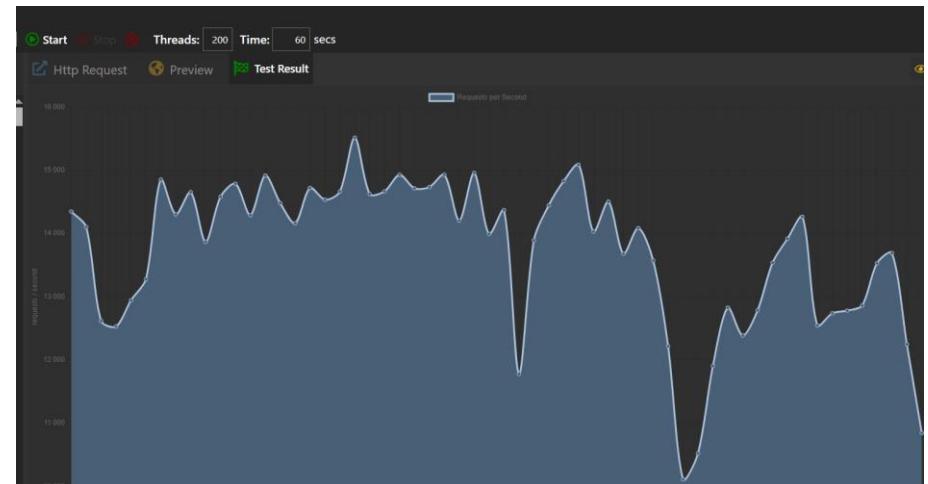
7.2.2. Test de performance

Ce test est réalisé après la containérisation de l'API, pour calibrer les ressources du container en fonction des besoins de concurrence de requêtes. Pour ce faire, j'ai réalisé un stress-test avec l'outil West Wind WebSurge¹⁹. Il m'a permis de réaliser environ 16000 requêtes par second, et ainsi constaté que la mémoire du container est correctement calibrée avec 256 Mo.

Statistiques du flux avec Portainer



Nombre de requête de connexion par second avec l'outil West Wind WebSurge



7.3. Déploiement

Suivant le cahier des charges, le déploiement de la base de données doit être réalisé sur le serveur de l'association. Pour l'API REST, et suivant la demande du PO, j'ai réalisé une image dockérisée, disponible sur mon DockerHub²⁰. Cette image docker permettra à l'association AMIO de déployer cette API sur leur solution de container interne, de type Docker desktop ou encore Rancher desktop.

¹⁹ <https://websurge.west-wind.com/>

²⁰ <https://hub.docker.com/repository/docker/marklhor/actifsmateriels/general>

7.3.1. Déploiement de l'API

Pour réaliser l'image de cette API, j'ai généré un Docker file avec Visual Studio permettant la fabrication de celle-ci.

Il comporte les données sur la version du Framework .NET, les projets devant être copiés, le type de publication, dans notre cas une « Release » pour la mise en production de l'API.

Pour réaliser l'image et la publier, j'ai employé l'outil de déploiement de Visual Studio. Celui-ci diffuse l'image fabriquée sur le Docker hub de mon choix. Cette image est accessible avec Docker desktop avec cette ligne de commande :

```
docker pull marklhor/actifsmateriels
```

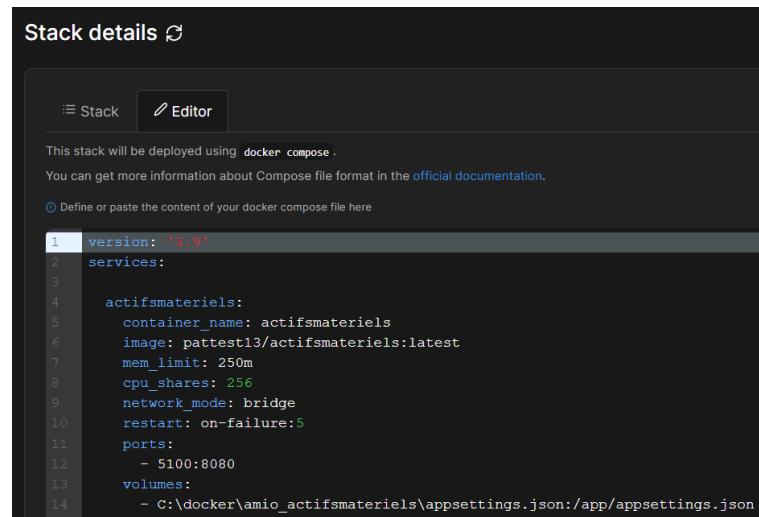
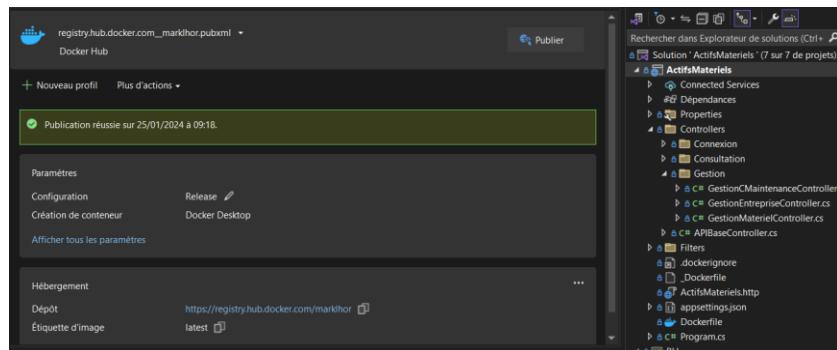
Avec l'outil Docker desktop et son plugin Portainer, j'ai rédigé une Stack permettant, avec la syntaxe Docker Compose, l'exécution de l'API dans un container sur le port de mon choix sur un serveur local.

```
FROM mcr.microsoft.com/dotnet/aspnet:8.0 AS base
USER app
WORKDIR /app
EXPOSE 8080

FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build
ARG BUILD_CONFIGURATION=Release
WORKDIR /src
COPY ["ActifsMateriels/ActifsMateriels.csproj", "ActifsMateriels/"]
COPY ["BLL/BLL.csproj", "BLL/"]
COPY ["DAL/DAL.csproj", "DAL/"]
COPY ["Domain/Domain.csproj", "Domain/"]
COPY ["Parameters/Parameters.csproj", "Parameters/"]
RUN dotnet restore "./ActifsMateriels./ActifsMateriels.csproj"
COPY . .
WORKDIR "/src/ActifsMateriels"
RUN dotnet build "./ActifsMateriels.csproj" -c $BUILD_CONFIGURATION -o /app/build

FROM build AS publish
ARG BUILD_CONFIGURATION=Release
RUN dotnet publish "./ActifsMateriels.csproj" -c $BUILD_CONFIGURATION -o /app/publish /p:UseAppHost=false

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "ActifsMateriels.dll"]
```



Le déploiement de l'image dockérisée permet, avec OpenAPI²¹, la visualisation des routes de l'API REST.

Celles-ci correspondent à la spécification du contrat entre l'API et le client Windows Forms, explicité au §4.3.2.

The screenshot shows the Swagger UI interface for a .NET application. The top navigation bar includes a back arrow, forward arrow, refresh button, and search icon. The URL in the address bar is 'localhost:5100/swagger/index.html'. The main content area is organized into sections:

- Connexion**: Contains three API endpoints:
 - POST /api/Connexion/loginSwagger**: Connexion à la page sagger pour lister les routes de l'API. Login 115@arnio.com et son mdp
 - GET /api/Connexion/testco**: DEV Route. Test de connexion
 - POST /api/Connexion**: Connexion à l'API pour les utilisateurs
- Consultation**: Contains two API endpoints:
 - GET /api/Consultation**: Fournit un Dto de consultation avec l'ensemble des actifs matériels, leurs catégories et si ils sont sous contrat de maintenance
 - GET /api/Consultation/{idCategorie}**: Fournit un Dto de consultation avec les actifs matériels en fonction de l'identifiant d'une catégorie
- GestionCMaintenance**: Contains one API endpoint:
 - GET /api/GestionCMaintenance**: Fournit un GestionCMaintenanceResponseDto avec l'ensemble des contrats de maintenance
- GestionEntreprise**: Contains one API endpoint:
 - GET /api/GestionEntreprise**: Fournit un GestionEntrepriseResponseDto avec l'ensemble des entreprise de maintenance
- GestionMateriel**: Contains several API endpoints:
 - GET /api/GestionMateriel**: Fournit un GestionMaterielResponseDto avec l'ensemble des actifs matériels, leurs catégories et si ils sont sous contrat de maintenance
 - GET /api/GestionMateriel/materiel/{idMat}**: fournit un Matériel en fonction de son identifiant et rétourne un GestionMaterielResponseDto
 - GET /api/GestionMateriel/categorie/{idCategorie}**: Fournit un GestionMaterielResponseDto avec les actifs matériels en fonction de l'identifiant d'une catégorie
 - POST /api/GestionMateriel/add**: Ajoute un matériel avec sa sélection de catégorie et rétourne un GestionMaterielResponseDto comprenant l'id et les catégories du nouveau matériel
 - PUT /api/GestionMateriel/update**: Mise à jour d'un matériel suivant son identifiant gestionMaterielRequestDto.OldIdMat et rétourne un GestionMaterielResponseDto comprenant l'id et l'identifiant modifié
 - PUT /api/GestionMateriel/archive**: Archivage d'un matériel suivant son identifiant gestionMaterielRequestDto.OldIdMat et rétourne un GestionMaterielResponseDto comprenant l'id et l'identifiant modifié
 - DELETE /api/GestionMateriel/delete/{idmat}**: Supprime un matériel suivant son identifiant gestionMaterielRequestDto.OldIdMat et rétourne un GestionMaterielResponseDto comprenant l'identifiant supprimé
 - GET /api/GestionMateriel/deleted/{idmat}**: Fournit un MatérielDeleted en fonction de son ancien identifiant dans la route
 - DELETE /api/GestionMateriel/delete/deleted/{idmat}**: Supprime un matériel détruit suivant son identifiant dans la route et rétourne un GestionMaterielResponseDto comprenant l'identifiant supprimé

²¹ <https://swagger.io/specification/>

8. Veille, effectuée sur les vulnérabilités de sécurité

Concernant la veille sur les vulnérabilités de la solution, je développe ici celle concernant les paramètres de [configurations](#). Paramètres concernent la création du token de connexion et la connexion à la base de données. Celle-ci pouvant être implémenté suivant le choix du type. Actuellement seul le type MariaDB est implémenté dans mon code source.

Pour sécuriser ces paramètres et ne pas les exposer dans l'image dockérisée, j'ai appliqué deux principes :

- Ignorer le fichier appsettings.json induit par le package NuGet [Microsoft.Extensions.Configuration](#) avec le fichier .docherignore
- Ignoré le fichier appsettings.json et appsettings.Integrations .json dans le gitignore
- Mapper un fichier appsettings.json en local, dans la stack de déploiement de l'image dockérisée.

Pour vérifier la ces deux points lors de la containérisation de l'image, j'ai ajouté la propriété "SettingType" dans les fichiers appsettings.json, du code source et dans celui en local. A cela j'ai modifié le DTO de réponse de connexion pour qu'il la porte. De plus, pour le choix du type de base de données, si ce type n'est pas implémenté, l'API renverra une exception.

Code source

Instanciation de la session suivant le paramètre
« DataBaseType »

```

appsettings.json
Schéma : https://json.schemastore.org/appsettings.json
1  "JWTToken": {
2    "JWTKey": "Henri Michaux Cette habitude disje je lai justement
3      gardée et jusquaujourdhui gardée secrète",
4    "JWTIssuer": "https://localhost:5100", //replace this with your
5      application url
6  },
7  "Logging": {
8    "LogLevel": {
9      "Default": "Information",
10     "Microsoft.AspNetCore": "Warning"
11   },
12   "AllowedHosts": "*",
13   "DBConnection": {
14     "ConnectionString": "server=lab004.2isa.org;port=33004;user id=root;password=3265Lab004; database=GESTION_TICKETS",
15     "ProviderName": "MySqlClient"
16   },
17   "SettingType": "inDockerImage",
18   "DataBaseType": "#" // 0=>MariaDB, 1=>SQLServer, 2=>PostgreSQL,
19   3=>Oracle
20 }

DAL.cs
45  I référence
46  public static IServiceCollection AddDAL(this IServiceCollection services, DbType dbType, Action<DALOptions> configure = null)
47  {
48    DALOptions options = new();
49    configure?.Invoke(options);
50    options.DBType = dbType;
51
52    //Register your services here
53    services.AddScoped<IDBSession>((services) =>
54    {
55      IConfiguration configuration = services.GetRequiredService< IConfiguration>();
56      string DBConnectionString = configuration.GetSection("DBConnection:ConnectionString").Value;
57      switch (options.DBType)
58      {
59        case DbType.MariaDB:
60          return new DBSessionMariaDB(options.DBConnectionString);
61        case DbType.SQLServer:
62          // return new DBSessionSQLServer(options.DBConnectionString);
63        case DbType.PostgreSQL:
64          // return new DBSessionPostgreSQL(options.DBConnectionString);
65        case DbType.Oracle:
66          // return new DBSessionOracle(options.DBConnectionString);
67        default:
68          //return new DBSessionMariaDB(options.DBConnectionString);
69          throw new SysException($"Le type de connexion à la base de données que
70 vous avez défini en paramètre, n'est pas implémenté. Veuillez
71 contacter le service informatique.");
72      }
73    });
74    services.AddTransient<IUOW, UOW>();
75  }
76

```

Docker compose avec [Portainer](#)

Réponse de l'API containérisée

The screenshot shows two panels. On the left, the 'Stack details' panel displays a Docker Compose file for a service named 'actifsmatériels'. The file includes definitions for container name, image, memory limit, CPU shares, network mode, restart policy, ports, and volumes. On the right, a 'Server response' panel shows a JSON object returned from the API endpoint `http://localhost:5100/api/Connexion`. The JSON object contains fields such as id, nomPrenom, role, isConnected, exception, accessToken, and settingType. A red arrow points to the 'settingType' field, which has a value of "outofDockerImage".

```
version: '3.9'
services:
  actifsmatériels:
    container_name: amio_api_actifsmatériels
    image: marklhor/actifsmatériels:latest
    mem_limit: 250m
    cpu_shares: 256
    network_mode: bridge
    restart: on-failure:5
    ports:
      - 5100:8080
    volumes:
      - c:/docker\AMIO_ActifsMateriel\appsettings\appsettin
```

```
{  
  "id": 96101,  
  "nomPrenom": "C. Patterson",  
  "role": 2,  
  "isConnected": true,  
  "exception": null,  
  "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ  
E4N2EyNzc1LWU20DMtNGZhYS1iMTE2LTA0NWUyM2IxMGMzYyIsImh0dHA  
nL3dzLzIwMDUvMDUvaWRLbnRpdkhvYxhaW1zL25hbwVpZGVudG1malV  
ZWIhc5taWNyb3NvZnQuY29tL3dzLzIwMDgvMDYvaWRLbnRpdkhvY2xha  
2Vyi10sImV4cCI6MTcwNjM1MDUzOCwiaXNzIjoiaHR0cHM6Ly9sb2Nhbg  
BzOib8vbG9jYWxob3N0bjUxMDAifQ.s9Q0iBJJVw8D1TpVLrgh8Et5fs2F  
  "settingType": "outofDockerImage" ←
```

Avec ces principes, j'ai permis la non-diffusion des informations de configurations. L'API ainsi ne peut être lancée sans le fichier appsettings.json, s'il n'est pas présent dans le répertoire local mappé dans la stack ; le container ne démarre pas. De plus, si les implémentations sont développées, la base de données pourra être modifiée à la volée. Un simple redémarrage du container de l'API REST permettra un changement de paramètres sur la base de données employée.

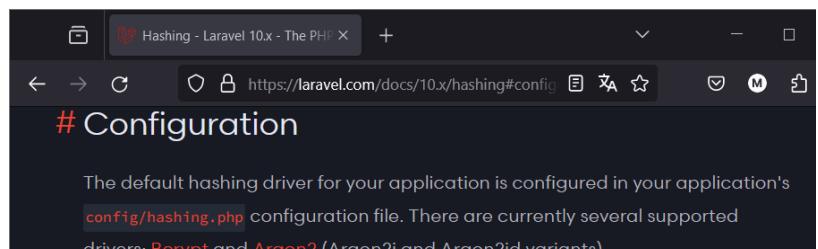
9. Description d'une situation de travail ayant nécessité une recherche

Le cryptage du mot de passe induit par la bibliothèque Fortify du Framework Laravel du premier jalon, fut ma première situation de travail ayant nécessité une recherche.

Ce cryptage du mot de passe de l'utilisateur et impliqué par dans le flux de connexion.

Dans un premier temps, j'ai identifié le type de cryptage du jalon un. Avec la documentation de [Laravel](#), j'ai pu identifier le fichier config/hashing.php, qui indique quel driver de hashage est employé par défaut.

La lecture de ce fichier de configuration de hashage indique que la bibliothèque Bcrypt est celle par défaut



```
hashing.php
1 <?php
2
3 return [
4
5     /*
6     | -----
7     | Default Hash Driver
8     | -----
9     |
10    | This option controls the default hash driver that will be used to hash
11    | passwords for your application. By default, the bcrypt algorithm is
12    | used; however, you remain free to modify this option if you wish.
13    |
14    | Supported: "bcrypt", "argon", "argon2id"
15    |
16    */
17
18    'driver' => 'bcrypt',
19]
```

```
public async Task<UserTransit> ConnexionUserAsync(string email, string password)
{
    var user = await _dbContext.User.GetByEmailAsync(email);
    if (user is null) throw new UnauthorizedAccessException(); ;
    bool verified = _bcrypService.Verify(password, user.password);
```

Dans mon code source, j'ai développé pour la connexion, un service Bcrypt qui permet la vérification de la concordance du mot de passe entré par l'utilisateur avec celui en base de données.

```
using BLL.Interface;
using Domain.Entities;

namespace BLL.Implementation;

public class BCrypService : IBCrypService
{
    public bool Verify(string expected, string tobechecked)
    {
        return BCrypt.Net.BCrypt.Verify(expected, tobechecked);
    }
}
```

10. Conclusion

L'objectif du premier jalon du projet Fil-Rouge de la formation CDA était de simplifier la gestion des incidents du parc informatique de l'association AMIO. Dans cette phase, j'ai développé une application client-serveur en utilisant le Framework .Net de Microsoft. Cette application vise à assurer le suivi de l'inventaire des actifs matériels et à faciliter la gestion de la maintenance en collaboration avec des fournisseurs externes. Conçu pour les professionnels de la maintenance informatique et de l'association, mon projet offre au service informatique une gestion globale de l'inventaire et un suivi pertinent de la maintenance et de leurs fournisseurs.

Pour satisfaire aux exigences formulées par AMIO dans le cahier des charges, visant une maintenance efficace et proactive pour les besoins futurs de l'application, les architectures retenues favorisent des évolutions simplifiées. À titre d'exemple, le changement de type de base de données peut être effectué via le fichier de configuration, tandis que le changement d'interface client peut être réalisé par la mise en œuvre d'une nouvelle implémentation de l'interface utilisateur.

AMIO a formulé une demande pour une architecture client-serveur, avec une spécification en n-tier pour le serveur. J'ai enrichi cette demande en optant pour une architecture MVC côté client, sans l'utilisation de bibliothèques supplémentaires. Cela permettra une mise en œuvre simplifiée des fonctionnalités supplémentaires mentionnées dans le cahier des charges comme « bonus », ainsi que celles à venir. Le choix du Framework .Net pour le client lourd et l'API a facilité le développement de la solution en utilisant un langage commun, le C#. Dans cette dernière étape du projet Fil-Rouge, j'ai conçu une application de bureau de type Windows Forms qui communique avec une API REST pour agir sur une base de données MariaDB.

Les attentes spécifiées qui ne figuraient pas dans le cahier des charges, mais qui découlaient de bonnes pratiques de développement logiciel trouvées sur des sites de confiance²², ont étoffé mon projet par rapport à mes réalisations antérieures. J'ai cherché à respecter les principes SOLIDE avec des spécifications détaillées, accompagnées d'une architecture, Modèle (Modèle de vue) Vue Contrôleur. Cependant, le processus de mise à jour de la vue par le modèle de vue nécessite des améliorations en raison de l'asynchronisme, provoquant une double mise à jour de la vue. Pour simplifier le développement, le passage d'une vue intermédiaire, où les fonctionnalités sont restreintes (boutons désactivés), à une vue où toutes les actions sont possibles, est géré par des booléen (_SwitchAdd, _SwitchUpdate) dans les flux de gestion matérielle. En conséquence, l'intégralité du code, depuis la réception des informations du modèle jusqu'à la mise à jour de la vue, est exécutée deux fois. Introduire un double verrouillage dans le contrôleur du client, avec un retour vers une vue intermédiaire, pourrait se rapprocher davantage d'une architecture asynchrone.

Mon approche de développement m'a permis d'intégrer de manière progressive les nouvelles connaissances acquises lors de ma formation, notamment les designs patterns, l'injection de dépendance, la dockerisation, et les informations issues de ma veille technologique. J'ai été confronté à divers défis tels que la compréhension des Claims²³, permettant de gérer les attributs d'autorisation du Token et sa définition, la gestion des exceptions au travers d'un filtre,

²² <https://philippe.developpez.com/articles/SOLIDdotNet/> - [https://openclassrooms.com/fr/courses/6810956-ecrirez-du-code-java-maintnable-avec-mvc-et-solid/6926857-decouvrez-les-bonnes-pratiques-de-programmation-avec-les-principes-solid](https://openclassrooms.com/fr/courses/6810956-ecrirez-du-code-java-maintenable-avec-mvc-et-solid/6926857-decouvrez-les-bonnes-pratiques-de-programmation-avec-les-principes-solid)

²³ <https://learn.microsoft.com/en-us/aspnet/core/security/authorization/claims?view=aspnetcore-8.0>

l'utilisation de la classe Mock dans les tests unitaires, et le strict respect des architectures. En suivant la démarche de développement basée sur le cycle de Deming, j'ai élaboré une solution applicative conviviale, efficace, évolutive et hautement maintenable. La maintenabilité de cette solution est vérifiable à travers les métriques du code calculées par Visual Studio. Ces métriques indiquent un indice de maintenabilité pour mes développements client et serveur, dépassant 36/100 et allant jusqu'à 94. Microsoft considère cet indice comme "correct" dès lors que le seuil de 20/100 est atteint (voir [annexe 7](#)).

Ce projet, réalisé dans le cadre de ma formation en tant que concepteur développeur d'applications, s'est avéré être une expérience extrêmement enrichissante. Il m'a donné l'opportunité de créer une solution complète client-api-base de données, en utilisant le Framework .Net de Microsoft avec deux architectures distinctes. La pertinence du projet et ses objectifs clairs ont constitué un guide précieux, me fournissant une direction bien définie. Les échanges fréquents avec les formateurs ont également joué un rôle crucial en orientant mes réflexions.

Cette expérience de conception et de développement logiciel a renforcé ma conviction quant à ma reconversion professionnelle, tout en me permettant d'acquérir de nouvelles compétences. Ces compétences se révèlent particulièrement pertinentes alors que je m'apprête à débuter la période d'application en entreprise à partir du 19 février 2024. Pendant cette période, je serai formé au métier d'analyse d'exploitation et rejoindrai une nouvelle équipe de développeurs. Notre principale activité consistera à moderniser les applications internes et à les déployer efficacement pour les clients de Ntico²⁴.

Cette étude logicielle m'a permis d'évaluer le temps de développement nécessaire pour une solution de ce type. Comme mentionné dans le [paragraphe 3.3](#), j'ai consacré 355 heures sur une période de 100 jours, ce qui équivaut à 51 jours-homme de travail à raison de 7 heures par jour, basé sur une semaine de travail de 35 heures. Cette activité s'est déroulée simultanément à l'acquisition de nouvelles compétences développées dans le cadre de ce projet.

Les compétences acquises, telles que la mise en œuvre de design patterns, l'architecture en couches, les principes des API REST, et la conception MVC du client, m'ont demandé un investissement en temps pour leur acquisition. En conséquence, si je devais envisager la réalisation d'un projet similaire, il me faudrait environ une centaine d'heures d'étude et de développement. Pour être plus précis, j'ai codé 800 lignes de code pour l'API, produisant 130 exécutables. Je serai capable de créer une solution de ce type, conforme aux spécifications (client Windows Forms, API REST, Token d'accès, avec une petite base de données MySQL), en un mois.

Je tiens à exprimer ma profonde gratitude envers mes formateurs qui m'ont accompagné tout au long de ce projet .NET, inscrit dans le cadre de ma mise en situation professionnelle pour l'obtention du titre professionnel de concepteur développeur d'applications. Leur expertise, leur dévouement et leur soutien ont été d'une importance inestimable. Grâce à leurs enseignements, j'ai pu consolider mes compétences et me préparer de manière efficace pour ce qui, je l'espère, sera le début d'une carrière prometteuse dans le développement d'applications, débutant par le stage à venir. Une fois de plus, je tiens à les remercier sincèrement pour le partage précieux de leurs connaissances, qui accompagne avec succès ma reconversion professionnelle.

²⁴ <https://www.ntico.com/>

11. Annexes

11.1. Annexe 1, QQOQCP

Qui ?	<p>Qui est concerné par ce projet ?</p> <ul style="list-style-type: none"> • Les professionnels de la maintenance informatique de l'organisation : <ul style="list-style-type: none"> ○ Les techniciens ○ Les administrateurs
	<p>Rationaliser la gestion de son parc informatique et en garantissant une maintenance efficace et proactive :</p>
Quelle est la problématique à résoudre ?	<ul style="list-style-type: none"> • Réaliser le suivi et l'inventaire des actifs matériels • Faciliter le suivi des contrats de maintenance avec les entreprises externe sur le matériel qu'elle fournit (matériel externe) • Bonus : gérer les contrats de maintenance avec des entreprises externes pour les matériels nécessitant une assistance externe <p>(Transfère matériel d'une entreprise à une autre, pas de suppression => archivage)</p>
Quoi ?	<p>Quelles sont les fonctionnalités demandées</p> <ul style="list-style-type: none"> • Consultation des données du matériel dans l'inventaire pour tous les utilisateurs authentifiés • Trie de la liste matérielle suivant une catégorie. • Modification des données matériels de l'inventaire uniquement pour le rôle ADMINISTRATEUR. • Consultation et modification des contrats sur les matériels externes uniquement pour le rôle ADMINISTRATEUR. • Bonus : <ul style="list-style-type: none"> ○ Modifier les contrats de maintenance avec des entreprises externes pour les matériels nécessitant une assistance externe ○ Modifier les entreprises externes pour les matériels nécessitant une assistance externe (Philippe Pallot au 13/10/23) • Authentification sécurisée pour garantir que seuls les utilisateurs autorisés ont accès aux informations sensibles. <p>Quelles sont les données à traiter</p> <ul style="list-style-type: none"> • Pour les matérielles : <ul style="list-style-type: none"> ○ Nom du matériel ○ Date de mise en service ○ Date de fin de garanti ○ Propriétaire actuel ○ Nom de la catégorie (Note : un matériel peut appartenir à plusieurs catégories) • Pour les contrats de maintenance : <ul style="list-style-type: none"> ○ Date de début de contrat ○ Date de fin de contrat ○ Date de dernière intervention ○ Date de prochaine intervention ○ Nom du contrat ○ Description du contrat ○ Entreprise chargée de remplir le contrat • Pour les entreprise prestataire des contrats de maintenance : <ul style="list-style-type: none"> ○ Nom de l'entreprise ○ Numéro de téléphone ○ Adresse courriel de contact

Où ?	Quels livrables sont demandés ?	<ul style="list-style-type: none"> • Un résumé du projet. • Un dossier projet. • Une présentation orale à partir d'un support de présentation type Powerpoint. • Une base de données, <ul style="list-style-type: none"> ◦ Un jeu de test cohérent et représentatif • Une application web service REST • Une application Windows Forms • Le déploiement sur un serveur de production fournis par l'équipe pédagogique.
		<p>Où l'application devra être installée</p> <ul style="list-style-type: none"> • Sur les postes des professionnels du service de maintenance informatique. <ul style="list-style-type: none"> ◦ Système d'exploitation : Windows 11
		<p>Où les livrables doivent être déposés ?</p> <ul style="list-style-type: none"> • Sur les postes des professionnels du service de maintenance informatique. <ul style="list-style-type: none"> ◦ Plateforme AMIO-FIT : Cours : Fil Rouge (2isa.eu)
	Ou le projet doit être déployé ?	Host xxx, HostName xxx, Port 22004, User xxx, ForwardAgent yes, Mot de passe : xxx
	Ou le projet doit être présenté ?	<ul style="list-style-type: none"> • Dans les locaux d'AMIO
	Début	<ul style="list-style-type: none"> • Mercredi 4 octobre 2023
	Évaluation individuelle	<ul style="list-style-type: none"> • Semaine du 12 février 2024
	Fin	<ul style="list-style-type: none"> • Vendredi 9 février 2024
	Quelles contraintes techniques ?	<p>Deux parties client, serveur):</p> <ul style="list-style-type: none"> • Client lourd développé sous la technologie Winforms avec le framework .NET et le langage C# • Une application web service REST • Code de qualité • Une main
	Comment ?	<p>Client lourd sur le poste de travail + API REST sur un serveur dédié</p> <p>L'interface de bureau sera développée sous la technologie WinForm de Microsoft.</p> <ul style="list-style-type: none"> • La BLL (Business Logic Layer) Client <p>La BLL client, devra utiliser la librairie standard .NET, elle contiendra toute la logique client et elle seule communiquera avec les API(s).</p>

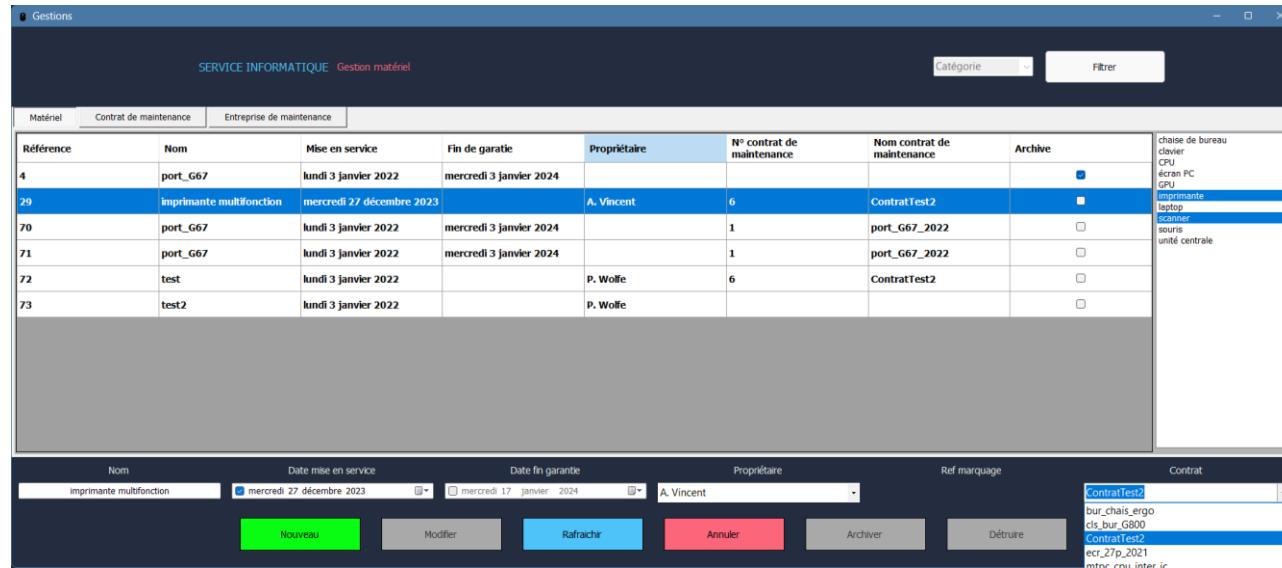
	<ul style="list-style-type: none"> • L'API REST Les composants serveur comme les API(s) devront être sous la technologie .Net 7. Elles permettront de faire une abstraction de notre code serveur, et de fournir aux clients un choix exhaustif de fonctionnalités. • BLL (Business Logic Layer) Serveur La BLL Serveur contiendra la logique du côté serveur et la cohérence des données. • DAL (Data Access Layer) La DAL quant à elle communiquera avec la base de données, elle pourra être développée avec le micro ORM « DAPPER » et le framework .Net 7.
Quels formats documentaires ?	<ul style="list-style-type: none"> • Tous les livrables devront être fournis au format numérique (PDF et Word) • Un résumé du projet : <ul style="list-style-type: none"> ◦ 200 à 250 mots ◦ Fourni au format Word (docx) et Pdf. • Le fichier devra être nommé : <ul style="list-style-type: none"> ◦ PRENOM_NOM_Projet_fil_rouge_Jalon2_Rresume.zip ◦ PRENOM_NOM_Projet_fil_rouge_Jalon2_Rresume.docx ◦ PRENOM_NOM_Projet_fil_rouge_Jalon2_Rresume.pdf • Un dossier projet fourni sous la forme d'une archive (zip). • Respecte du plan type : <ol style="list-style-type: none"> 1. Liste des compétences du référentiel qui sont couvertes par le projet 2. Cahier des charges, expression des besoins 3. Spécifications fonctionnelles du projet 4. Spécifications techniques du projet, élaborées par le candidat, y compris pour la sécurité 5. Réalisations du candidat comportant les extraits de code les plus significatifs, et en les argumentant, y compris pour la sécurité 6. Plan de tests et de déploiement 7. Description de la veille, effectuée par le candidat durant le projet, sur les vulnérabilités de sécurité 8. Description d'une situation de travail ayant nécessité une recherche, effectuée par le candidat durant le projet, à partir de sites francophones ou anglophones • Support de présentation orale au format type Powerpoint. • Description de la veille sur les vulnérabilités de sécurité : <ul style="list-style-type: none"> ◦ Décrire comment vous avez effectué la veille de sécurité : les sites et les mots clés utilisées et indiquer les vulnérabilités trouvées et éventuellement les failles potentielles corrigées.
Quelles sont les contraintes rédactionnelles ?	<ul style="list-style-type: none"> • Description de la situation de travail, ayant nécessité une recherche basée sur un ou des sites anglophones. La description doit concerner un problème technique ou une nouvelle fonctionnalité mise en œuvre, dans le cadre du projet. : <ul style="list-style-type: none"> ◦ Décrire le besoin d'information, et indiquer comment vous avez effectué la recherche : les mots clés des recherches utilisés et la liste des sites retournés. ◦ Pour finir précisez les critères de sélection du (ou des) site(s) et indiquer la solution trouvée et si elle a pu être mise en œuvre. • Longueur du dossier de projet hors annexes : 30 pages, soit environ 48750 caractères espaces non compris.
Quelles sont les contraintes temporelles de la présentation orale ?	<ul style="list-style-type: none"> • La présentation orale se déroulera en deux parties : <ol style="list-style-type: none"> 1. 40 minutes pour la présentation de la réalisation du projet 2. 30 minutes d'entretien technique avec le jury.

	<ul style="list-style-type: none">• Un support de présentation• Une présentation professionnelle• Objectifs et les compétences visées annoncées.• Contexte situé.
Quelles sont les contraintes de la présentation orale ?	<ul style="list-style-type: none">• Reformulation des contours du projet et de ses objectifs fonctionnels par type d'utilisateurs• Détail de l'architecture employée• Mise en évidence des principales étapes du travail réalisé• Présentation des résultats les plus pertinents pour démontrer les compétences en jeu• Argumenterez les choix• Exposerez les difficultés rencontrées et les solutions apportées• Présentez l'organisation de la gestion du projet• Conclure en faisant le point sur l'état d'avancement et dresser le bilan à cette étape
Pourquoi ?	<p>Quels sont les buts de l'application ?</p> <ul style="list-style-type: none">• Rationnaliser la gestion du parc informatique• Garantir et améliorer l'efficacité dans sa gestion• Permettre une maintenance proactive des solutions applicatives

11.2. Annexe 2, Phase d'ajout d'un actif matériel

Etape 1 :

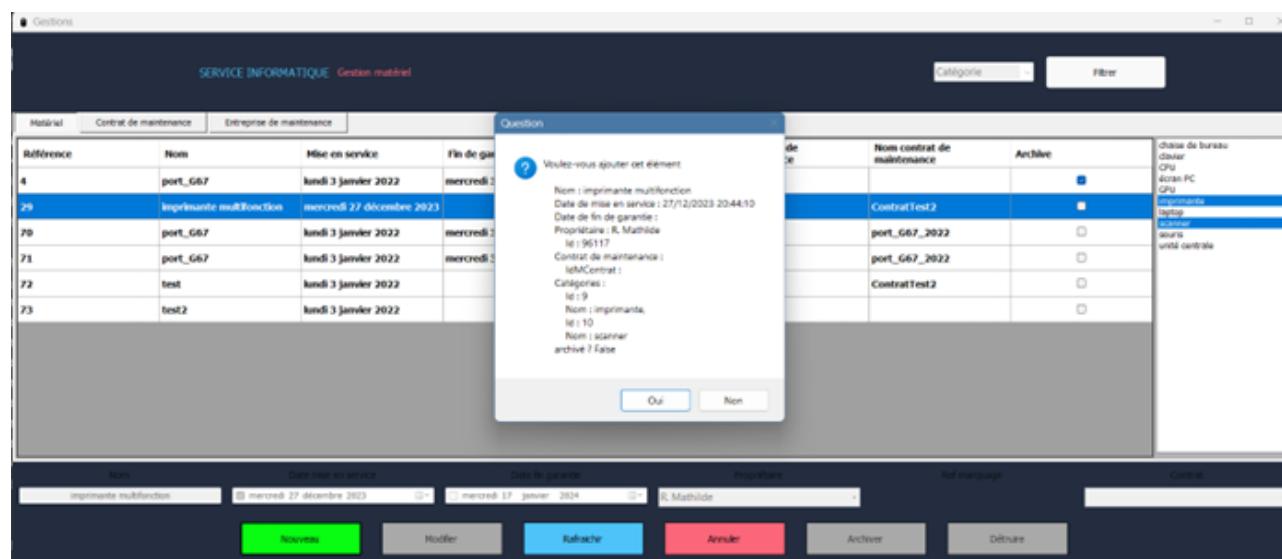
Cliquer sur le bouton « Nouveau » pour activer le mode « ajout matériel »



Etape 2 :

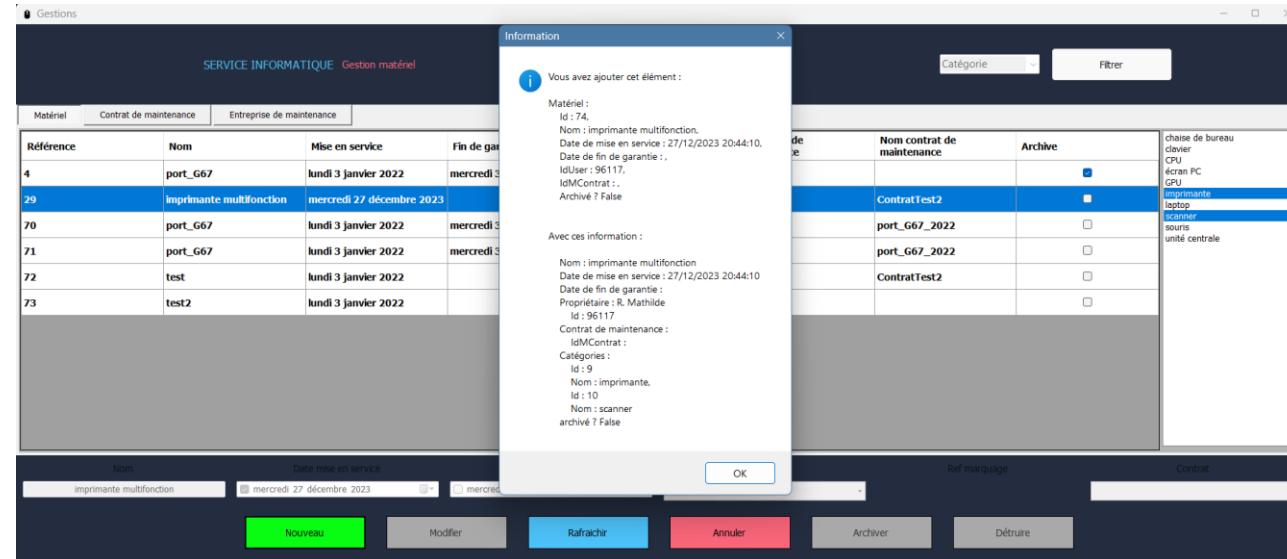
Compléter les données du nouvel actif et cliquer sur « Nouveau ».

Une confirmation vous sera demandée.

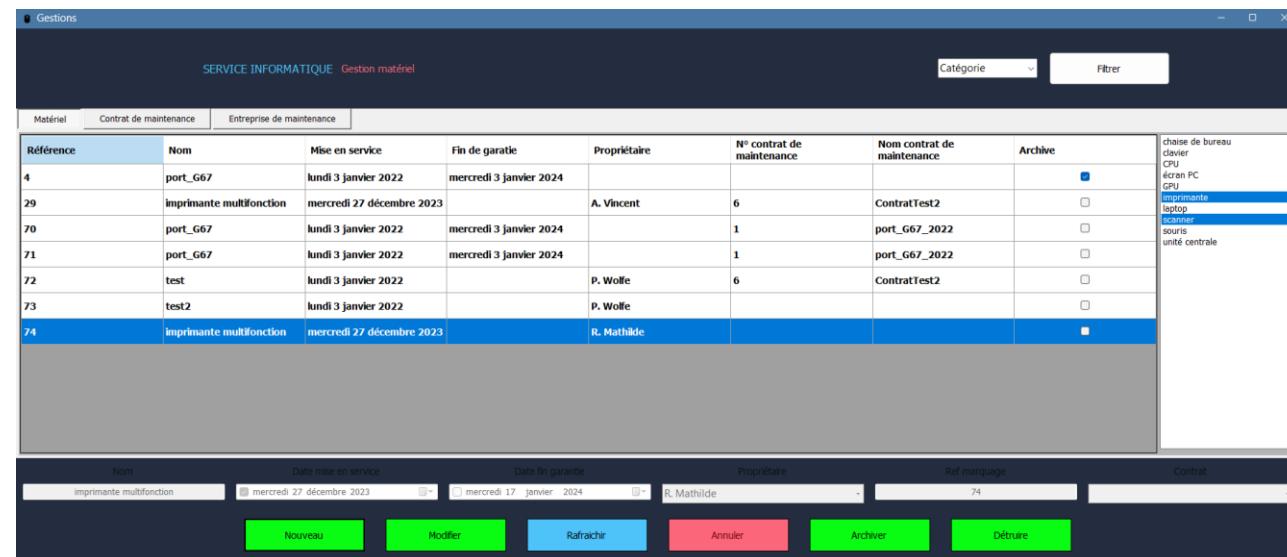


Etape 3 :

Une confirmation de l'ajout vous sera affichée, avec vos données d'entrées, et celles présentes dans la base de données du nouvel actif matériel ajouté.

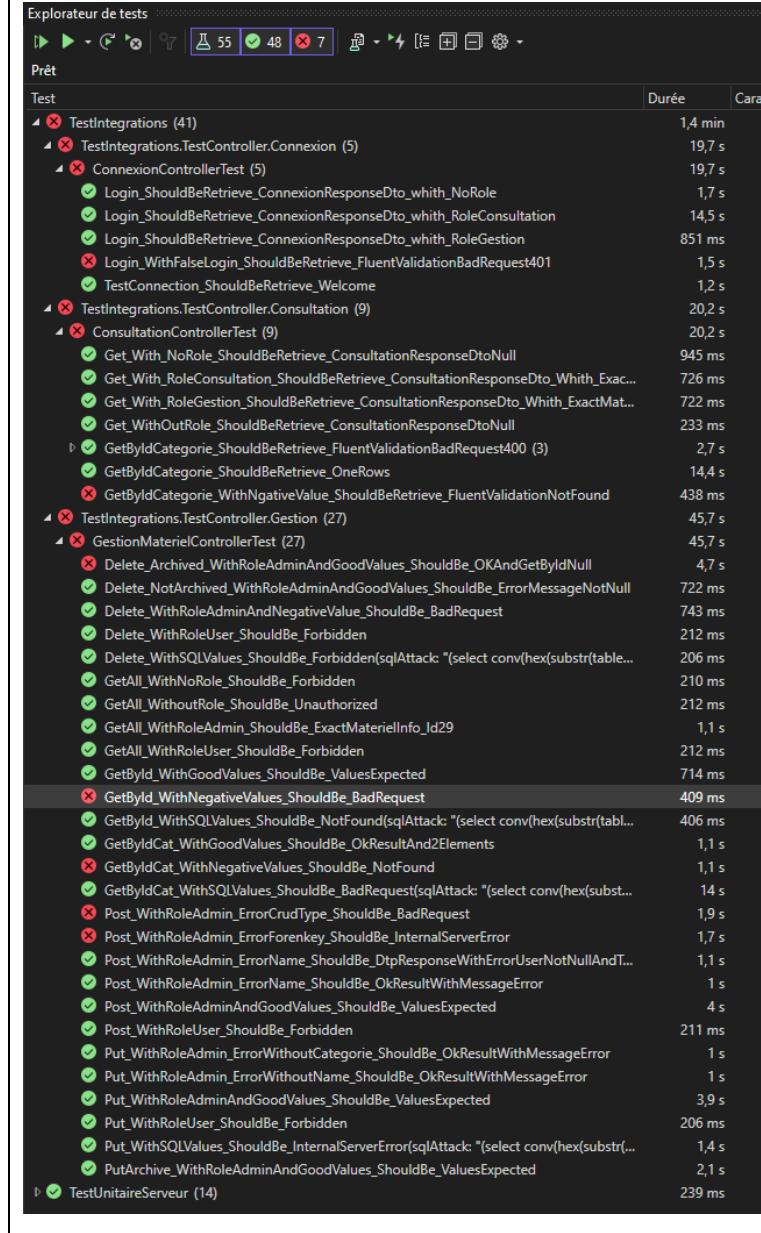
**Etape 4 :**

Vous serez repositionné sur ce nouveau matériel

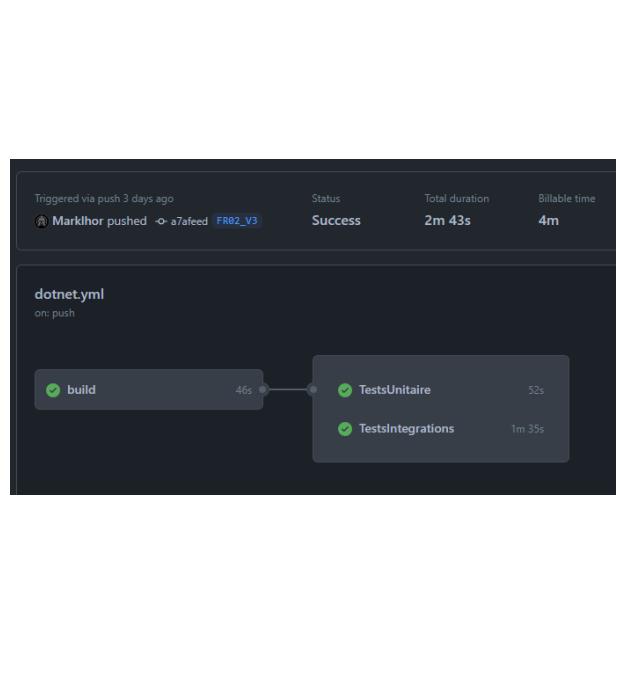


11.3. Annexe 3, passage de .NET 6 à 8

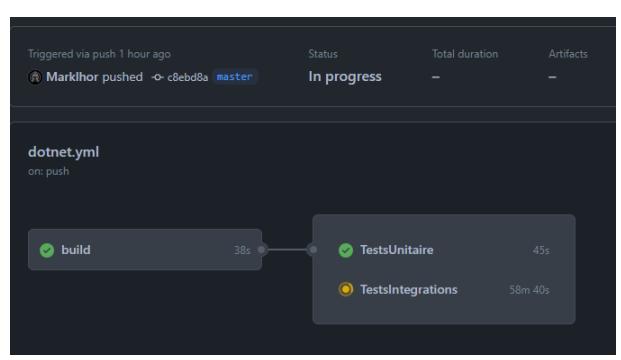
Régression dans les tests d'intégrations par suite de la mise-à-jour du Framework, le 18/01/2024



The screenshot shows the Visual Studio Test Explorer interface. It lists 41 failed tests under the 'Test' category. The tests are categorized into several groups: 'TestIntegrations' (41), 'TestIntegrations.TestController.Connexion' (5), 'ConnexionControllerTest' (5), 'TestIntegrations.TestController.Consultation' (9), 'ConsultationControllerTest' (9), 'TestIntegrations.TestController.Gestion' (27), and 'GestionMaterielControllerTest' (27). Most tests are marked with a red 'X' indicating failure, while some are marked with a green checkmark.

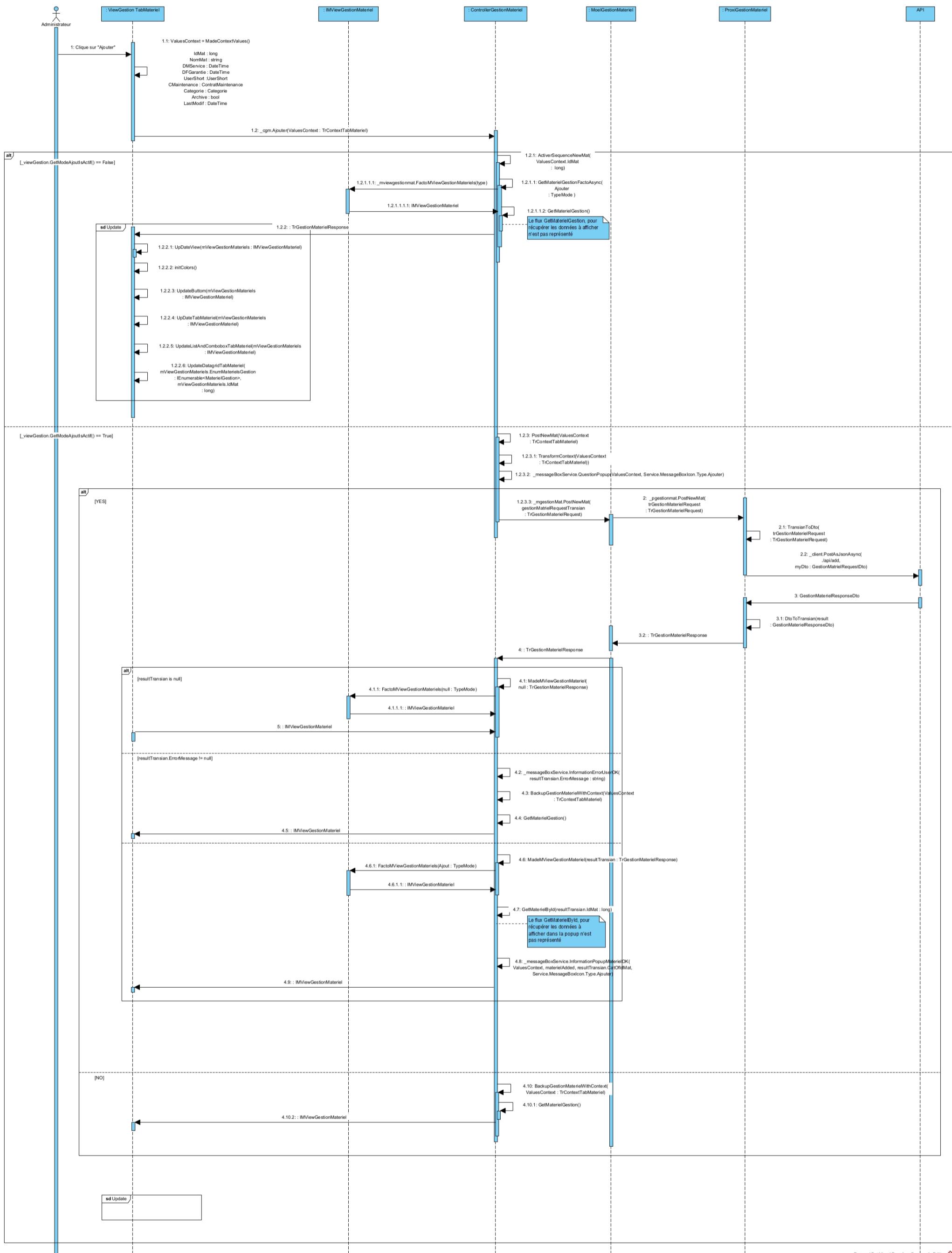


The screenshot shows a CI pipeline named 'dotnet.yml' triggered via push 3 days ago. The status is 'Success'. The total duration was 2m 43s and the billable time was 4m. The pipeline consists of three steps: 'build', 'TestsUnitaire' (status: success, duration: 52s), and 'TestsIntegrations' (status: success, duration: 1m 35s).



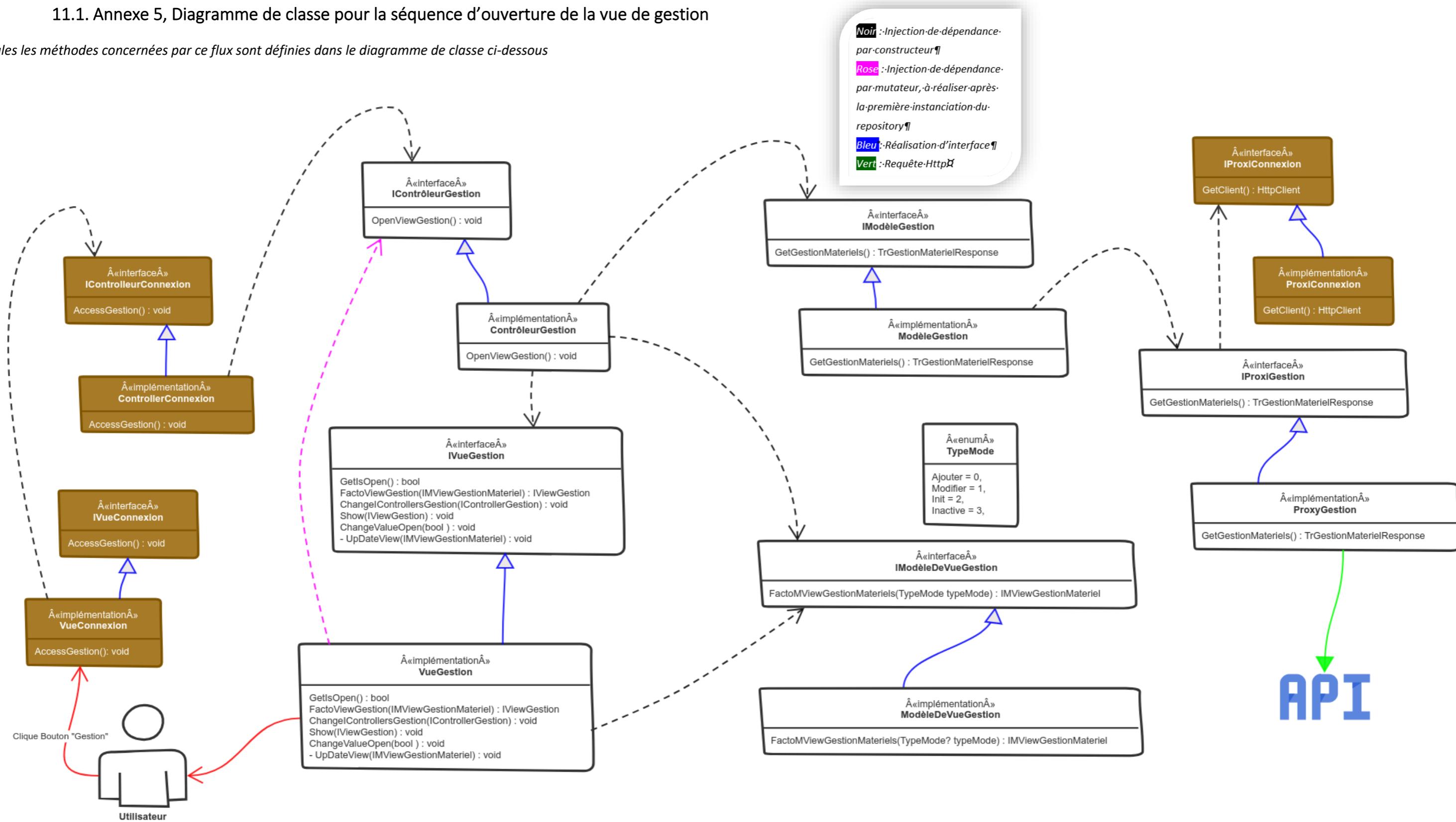
The screenshot shows a CI pipeline named 'dotnet.yml' triggered via push 1 hour ago. The status is 'In progress'. The total duration is '-' and there are no artifacts. The pipeline consists of three steps: 'build' (duration: 45s), 'TestsUnitaire' (status: success, duration: 45s), and 'TestsIntegrations' (status: in progress, duration: 58m 40s).

11.4. Annexe 4, diagramme de séquence du flux d'ajout du client



11.1. Annexe 5, Diagramme de classe pour la séquence d'ouverture de la vue de gestion

Seules les méthodes concernées par ce flux sont définies dans le diagramme de classe ci-dessous



11.2. Annexe 6, Exemple de test unitaire d'analyse du Token access Jwt

```
public async void
ConnexionUser_With_RoleGestion_Should_UserTransitWithRoleAdminAndUserInToken()
{
    // Arrange (arranger)
    // je simule unit of work et le service de de BCrypt
    IUOW _dbContext = Mock.Of<IUOW>();
    IBCrypService _bCrypService = Mock.Of<IBCrypService>();

    // je simule le fichier de configuration appsettings.json par un dictionary
    var inMemorySettings = new Dictionary<string, string> {
        {"JwToken:JwtKey", "Henri Michaux : Cette habitude, dis-je, je l'ai justement
gardée, et jusqu'aujourd'hui gardée secrète"},
        {"JwToken:JwtIssuer", "https://localhost:5294"},
        {"JwToken:JwtExpireDays", "1" },
    };
    // j'injecte cette nouvelle implémentation
    IConfiguration _configuration = new
ConfigurationBuilder().AddInMemoryCollection(inMemorySettings).Build();

    // je définie les valeurs d'entrée de la fonction à tester
    var emailInputValue = "115@amio.com";
    var passwordInputValue = "+1Mot@Passe";
    // je définie le résultat attendu, utilisateur correspondant au couple email+mdp
    var userExpected = new User()
    {
        id = 96101,
        name = "Patterson",
        firstname = "Cooper",
        email = "115@amio.com",
        password = "+1Mot@Passe",
        idSIRole = 2, // => Excepted value "Admin", "User"
    };
    // je simule le résultat attendu dans le token
    var resultExcepted = new List<string>() { "Admin", "User" }; //TODO correct ??? ou
l'obtenir avec le userExpected et décomposer le token ???

    // je simule
    Mock.Get(_dbContext).Setup(_bd =>
    _bd.User.GetByEmailAsync(emailInputValue)).ReturnsAsync(userExpected);
    var a = Mock.Get(_bCrypService).Setup(_db => _db.Verify(passwordInputValue,
userExpected.password)).Returns(true);

    // construction de l'instance à tester
    var connexionService = new BLLConnexionService(_dbContext, _configuration,
_bCrypService);

    // Act (réaliser)
```

```
var resultActual = await connexionService.ConnexionUserAsync(emailInputValue,
passwordInputValue);

    // Asset (vérifier)
    // je récupère la liste des rôles dans le Token par la classe
JwtSecurityTokenHandler https://learn.microsoft.com/en-
us/dotnet/api/system.identitymodel.tokens.jwt.jwtsecuritytokenhandler?view=msal-web-
dotnet-latest
    var ActualToken = resultActual.AccessToken;
    var Handler = new JwtSecurityTokenHandler();
    var HandlerToken = Handler.ReadJwtToken(ActualToken) as JwtSecurityToken;
    var RolesActualToken = (HandlerToken.Claims.Where(c =>
c.Type.Contains("role")).ToList()).Select(c => c.Value).ToList();
    // func lambda pour tester les correspondance entre Actual et Excepted
    var test = () =>
{
    bool result = false;
    int compt = 0;
    foreach (var roleActual in RolesActualToken)
    {
        if (!resultExcepted.Contains(roleActual))
            result = false;
        else
        {
            result = true;
            compt++;
        }
    }
    if (compt != RolesActualToken.Count && RolesActualToken.Count != resultExcepted.Count)
        return false;
    else return true;
};
Assert.True(test());
}
```

11.3. Annexe 7, Métriques de la solution client-serveur

Les métriques mesurent la complexité et la facilité de maintenance du code.

Voir <https://learn.microsoft.com/fr-fr/visualstudio/code-quality/code-metrics-values?view=vs-2022>

Ci-contre les explications des indicateurs dits métriques.

Je constate pour ma solution :

- Une différence trop importante entre le nombre de ligne de code et ceux exécutables. Je dois donc coder que l'utile et éviter l'inflation de ligne inutilisées
- Un couplage tout important, au-dessus du 9 recommandé par Microsoft pour des méthodes clés de mon code source. Exemple la OpenViewGestion() du client. Je devrai pour mes futurs développements, porter une forte attention à cet indicateur.
- Pour la valeur d'héritage, aucune valeur de référence n'existe. Elle doit être le plus faible possible afin de minimiser les erreurs. Dans mon cas cette valeur oscille entre 1 et 3 par méthode.
- La complexité cyclonique qui mesure le nombre de décisions prises dans mon code source, doit être faible. Microsoft indique qu'elle ne doit pas dépassée les 10. Dans le cas de la fonction PutGestionMaterielsUpdateAsync(GestionMatrielRequestDto) de l'API, cet indicateur est à 24. Je devrai pour mes futurs développements, porter une forte attention à cet indicateur.
- L'indice de maintenabilité pour chaque méthode exprime une bonne maintenabilité pour l'ensemble du projet

- **Index de maintenabilité** : calcule une valeur d'index comprise entre 0 et 100 qui représente la facilité relative de maintenance du code. Une valeur élevée signifie une meilleure maintenabilité. Les évaluations codées en couleur peuvent être utilisées pour identifier rapidement les points de problème dans votre code. Une évaluation verte est comprise entre 20 et 100 et indique que le code a une bonne maintenabilité. Une évaluation jaune est comprise entre 10 et 9 et indique que le code est modérément maintenable. Une évaluation rouge est une évaluation comprise entre 0 et 9 et indique une faible facilité de maintenance. Pour plus d'informations, consultez [Plage et signification de l'indice de maintenabilité](#).
- **Complexité cyclomatique** : mesure la complexité structurelle du code. Il est créé en calculant le nombre de chemins de code différents dans le flux du programme. Un programme qui a un flux de contrôle complexe nécessite davantage de tests pour obtenir une bonne couverture du code et est moins maintenable. Pour plus d'informations, consultez [Complexité cyclomatique](#).
- **Profondeur de l'héritage** : indique le nombre de classes différentes qui héritent les unes des autres, jusqu'à la classe de base. La profondeur de l'héritage est similaire au couplage de classes, car une modification d'une classe de base peut affecter l'une de ses classes héritées. Plus ce nombre est élevé, plus l'héritage est profond et plus le potentiel de modifications de classe de base peut entraîner un changement cassant. Pour la profondeur de l'héritage, une valeur faible est bonne et une valeur élevée est incorrecte. Pour plus d'informations, consultez [Profondeur de l'héritage](#).
- **Couplage de classes** : mesure le couplage avec des classes uniques via des paramètres, des variables locales, des types de retour, des appels de méthodes, des instantiations génériques ou de modèles, des classes de base, des implémentations d'interfaces, des champs définis sur des types externes et une décoration d'attributs. Une bonne conception logicielle détermine que les types et méthodes doivent avoir une cohésion élevée et un couplage faible. Le couplage élevé indique une conception difficile à réutiliser et à maintenir en raison de ses nombreuses interdépendances sur d'autres types. Pour plus d'informations, consultez [Couplage de classes](#).
- **Lignes de code source** : indique le nombre exact de lignes de code source présentes dans votre fichier source, y compris les lignes vides. Cette métrique est disponible à partir de Visual Studio 2019 version 16.4 et Microsoft.CodeAnalysis.Metrics (2.9.5).
- **Lignes de code exécutable** : indique le nombre approximatif de lignes de code exécutables ou d'opérations. Il s'agit d'un nombre d'opérations dans le code exécutable. Cette métrique est disponible à partir de Visual Studio 2019 version 16.4 et Microsoft.CodeAnalysis.Metrics (2.9.5). La valeur est généralement proche de la mesure précédente, **Lignes de code**, qui est la métrique basée sur l'instruction MSIL utilisée en mode hérité.

Métriques du client Windows Forms

		Indice de maintenabilité	Complexité cyclotique	Profondeur d'héritage	Couplage de classe	Lignes de code source	Lignes de code exécutée
Résultats de la métrique du code							
Filter : Aucun	Min : 0	Max : 100					
Hierarchie ▲							
Domain (Debug)	93	377	2	38	848	7396	
GestionActifsMateriel (Debug)	89	936	7	213			
ApplicationConfiguration	83	1	1	3	21		
GestionActifsMaterielObjTransf.Gestion.GestionMateriel	100	11	1	6	14		
GestionActifsMateriel.Properties	84	24	3	17	221		
GestionMaterielExt	76	159	7	112	1924		
GestionMaterielExt.Controller.Connexion.Implementation	64	6	1	15	123		
GestionMaterielExt.Controller.Connexion.Interface	100	3	0	3	18		
GestionMaterielExt.Controller.Consultation.Implementation	59	9	1	15	178		
GestionMaterielExt.Controllers.Gestion.Implementation	100	3	0	3	9		
ControllerGestionMaintenance	67	60	1	39	644		
ControllerGestionMaintenance	70	4	1	14	51		
ControllerGestionEnterprise	70	4	1	14	51		
ControllerGestionMateriel	81	52	1	33	536		
GestionMateriel : ModelGestionMateriel	100	0	1	1			
mvviewGestionMat : IViewGestionMateriel	100	0	1	1			
messageBoxService : IMessageBoxService	100	0	1	1			
rgnm : KontrollerGestionMaintenance	100	0	1	1			
rgte : IControllerGestionEnterprise	100	0	1	1			
viewGestion : ViewGestion	100	0	1	1			
ControllerGestionMateriel(IModelGestionMateriel, IViewGestion)	69	1	6	9			
OpenViewGestion() : Task	56	4	17	33			
GetMaterial(Gestion) : Task<IViewGestionMateriel>	67	1	9	18			
GetMaterial(GestionCategory) : Task<IViewGestionMateriel>	58	3	14	29			
GetMaterial(IdSyncing) : Task<Material>	78	2	3	6			
Ajouter(TContextTabMateriel) : Task<IViewGestionMateriel>	69	4	9	22			
ActiveSequenceNewMat(IdSync) : Task<IMViewGestionMateriel>	75	1	3	12			
PostNewMatInContextTabMateriel() : Task<IMViewGestionMateriel>	55	4	15	52			
Modifier(TContextTabMateriel) : Task<IMViewGestionMateriel>	70	2	9	21			
ModifierSeqInContextTabMateriel : Task<IMViewGestionMateriel>	73	3	5	9			
ModifierSeqInContextTabMateriel : Task<IMViewGestionMateriel>	55	4	15	45			
Archiver(TContextTabMateriel) : Task<IMViewGestionMateriel>	52	4	16	57			
Supprimer(TContextTabMateriel) : Task<IMViewGestionMateriel>	52	4	16	68			
GetMaterialSupprimeById(IdSync) : Task<Material>	93	1	3	4			
GetMaterialGestionFactoAsync(TypeMode) : Task<IMViewGestion>	61	2	14	32			
MadeIMViewGestionMateriel(TrGestionMaterielResponse, Nullable<IMaterial>) : Task	56	2	12	35			
MadeUserShort(TContextTabMateriel) : List<UserShort>	63	2	7	14			
TransformContent(TContextTabMateriel) : TrGestionMaterielResult	62	5	12	29			
BackupGestionMaterielWithContext(TContextTabMateriel) : Task<IMaterial>	68	3	4	9			
GestionMaterielExt.Controller.Gestion.Interface	100	11	0	7	84		
GestionMaterielExt.Emlt.Controller	76	10	1	13	194		
GestionMaterielExt.Emlt.Factory	91	15	2	7	90		
GestionMaterielExt.Emlt.Model	68	7	1	20	241		
GestionMaterielExt.Emlt.Test	71	2	1	2	17		
GestionMaterielExt.Model	82	18	1	4	56		

		Indice de maintenabilité	Complexité cyclotique	Profondeur d'héritage	Couplage de classe	Lignes de code source	Lignes de code exécutée
Résultats de la métrique du code							
Filter : Indice de maintenabilité	Min : 0	Max : 100					
Hierarchie ▲							
BLLExtension.DALOptionsDelegate	100	1	1	1	1	1	1
BLLImplementation	66	97	1	1	68	802	
BCryptService	94	1	1	1	3	7	
BLLConnectionService	64	11	1	1	31	127	
BLLConsultationService	69	3	1	1	12	67	
BLLGestionMaintenanceService	60	6	1	1	18	55	
BLLGestionEnterpriseService	60	5	1	1	16	53	
BLLGestionMaterielService	54	71	1	1	35	481	
dbContext : IUnitOfWork	100	0	1	1			
BLLGestionMaterielService(UOW)	96	1	1	1			
GetGestionMaterielSync() : Task<IGestionMaterielResponseDto>	100	1	1	1			
GetGestionMaterielByCatAsycn(IdSync) : Task<IGestionMaterielResponseDto>	57	6	1	1	13	34	
GetGestionMaterielByMatAsycn(IdSync) : Task<IGestionMaterielResponseDto>	67	2	1	1	9	16	
GetMaterielByAync(IdSync) : Task<IMaterial>	74	2	1	1	6	12	
FuncGetAllGestionMateriel() : Task<IGestionMaterielResponseDto>	46	9	1	1	25	54	
PostGestionMaterielAddAsync(GestionMaterielRequestDto) : Task<IGestionMaterielResponseDto>	45	11	1	1	19	74	
PutGestionMaterielUpdateAsync(GestionMaterielRequestDto) : Task<IGestionMaterielResponseDto>	36	24	1	1	20	127	
PutGestionMaterielArchiveAsync(GestionMaterielRequestDto) : Task<IGestionMaterielResponseDto>	57	5	1	1	8	52	
DeleteGestionMaterielAsync(IdSync) : Task<IGestionMaterielResponseDto>	49	7	1	1	17	75	
GetMatDeletedAsync(IdSync) : Task<IMaterialDefault>	83	1	1	1	4	5	
DeleteMatDeletedGestionMaterielAsync(IdSync) : Task<bool>	81	2	1	1	3	5	
BLLInterface	100	16	0	1	9	125	
IBCryptService	100	1	0	1	0	4	
IBLConnectionService	100	1	0	1	2	9	
IBLCrossTableService	100	2	0	1	2	6	
IBLGestionMaintenanceService	100	1	0	1	2	4	
IBLGestionEnterpriseService	100	1	0	1	2	5	
IBLGestionMaterielService	100	10	0	1	5	85	
DAL (Debug)	89	134	1	1	58	805	
DAL	88	50	1	1	35	187	
DALSession	59	7	1	1	14	53	
AddDAL(this IServiceCollection, DbType, Action<DALOptions>) : IServiceCollection	59	7	1	1	14	29	
DALOptions	96	5	1	1	1	13	
DbType	100	1	1	1	0	19	
IUnitOfWork	100	11	0	1	9	34	
db : IDBSession	88	26	1	1	23	79	
User : IUserRepository	100	0	1	1	1	1	
MaterielInfos : IMaterielInfosRepository	100	2	1	1	1	1	
Category : ICategoryRepository	100	2	1	1	1	1	
Materiel : IMaterielRepository	100	2	1	1	1	1	
Enterprise : IEnterpriseRepository	100	2	1	1	1	1	
ContratMaintenance : IContratMaintenanceRepository	100	2	1	1	1	1	
MaterielCategory : IMaterielCategoryRepository	100	2	1	1	1	1	
MaterielDefault : IMaterielDefaultRepository	100	2	1	1	1	1	