Rapport d'élève ingénieur
Formation initiale sous statut d'apprenti

1ère année

InterOp-Box

Application web modulaire pour l'IT

Présenté par Emilien ROUKINE

Tuteur ISIMA :
Claude MAZEL

Période :
Septembre 2025 — Juin 2026

Maître d'apprentissage:
Guillaume GUÉRIN

Soutenance :
24 juin 2026

ISIMA — Campus des Cézeaux · 1 rue de la Chebarde, TSA 60026 · 63178 Aubière CEDEX

Assistance IA : limitée

Remerciements

Je tiens tout d'abord à remercier l'entreprise Limagrain de m'avoir accueilli et de m'avoir fait confiance tout au long de cette première année d'alternance, dans le cadre de ma formation d'ingénieur à l'ISIMA.

Je remercie chaleureusement mon maître d'apprentissage, Guillaume Guérin, pour le temps qu'il m'a accordé, son écoute et son encadrement quotidien. Son expertise technique et ses précieux conseils m'ont permis de progresser de manière significative lors de mes missions. J'adresse également un grand merci à Anass pour son aide et pour nos intéressants échanges tout au long de l'année.

Enfin, je remercie mon tuteur académique à l'ISIMA, Monsieur Claude Mazel, pour son suivi attentif, sa disponibilité.

Table des matières

1 Introduction 7
2 Contexte et objectifs de l'étude 7
2.1 Présentation de l'entreprise 7
2.1.1 Origines, statut et identité 7
2.1.2 Une organisation en Business Lines 7
2.1.3 Chiffres clés (2024-2025) 8
2.1.4 Recherche et développement 9
2.1.5 Démarche environnementale et RSE 9
2.1.6 Équipe Intégration & Interopérabilité 9
2.2 Rappel du sujet 10
2.3 Analyse du problème 11
3 Conception d'une solution 11
3.1 Choix technologiques 12
3.1.1 Frontend — Angular et Tailwind CSS 12
3.1.2 Backend — Python et FastAPI 12
3.2 Architecture logicielle 12
3.2.1 Organisation du backend 13
3.2.2 Organisation du frontend 13
3.2.3 Système de modules dynamiques 14
4 Authentification 14
4.1 Flux d'authentification avec Microsoft Entra ID 15
4.2 Gestion des sessions 15
4.3 Rôles et permissions 16
5 Intégration d'un système externe : le cas GoAnywhere MFT 16
5.1 Première tentative : rétro-ingénierie de l'API privée avec Spectral 17
5.2 Accès direct à la base de données 17
5.3 Scraping comportemental : simuler un utilisateur réel 18
5.4 Synthèse des approches 18
6 Conclusion 19
7 Références bibliographiques 19
8 Webographie 20
9 Annexes 21
9.1 Annexe A — Page d'accueil d'InterOp-Box 21
9.2 Annexe B — Flux d'authentification avec Microsoft Entra ID 22
9.3 Annexe C — Scraping comportemental : séquence d'appels 23

Table des figures

Fig. 1 Répartition du chiffre d'affaires consolidé de Limagrain (2024) 8
Fig. 2 Principaux engagements environnementaux et sociaux de Limagrain (2024) 9
Fig. 3 Organigramme — Position de l'équipe Intégration & Interopérabilité 10
Fig. 4 Architecture en couches d'InterOp-Box 13
Fig. 5 Page d'accueil d'InterOp-Box — liste des modules actifs 21
Fig. 6 Flux d'authentification OAuth 2.0 avec Microsoft Entra ID 22
Fig. 7 Séquence d'appels HTTP lors de la création d'un compte via scraping comportemental 23

Lexique

API (*Application Programming Interface*) : interface logicielle qui définit un contrat de communication entre deux systèmes, en exposant un ensemble d'opérations accessibles à distance sans exposer le code interne.

Endpoint : point d'accès d'une API, identifié par une URL et une méthode HTTP (GET, POST, etc.), auquel on envoie une requête pour déclencher une opération ou récupérer des données.

Entra ID : service d'annuaire et de gestion des identités de Microsoft (anciennement Azure Active Directory), utilisé par Limagrain comme fournisseur d'identité central pour l'ensemble des applications internes.

MFA (*Multi-Factor Authentication*) : authentification multifacteur, mécanisme qui exige une preuve d'identité supplémentaire au mot de passe (code à usage unique, notification sur un appareil de confiance, etc.).

Scraping comportemental : technique consistant à simuler par programme les requêtes HTTP qu'un utilisateur humain produirait en naviguant dans une interface web, afin d'interagir avec un système ne disposant pas d'API exploitable.

Singleton : patron de conception logicielle garantissant qu'une classe ne possède qu'une seule instance partagée dans toute l'application, utilisé notamment pour les services de configuration ou de connexion.

SQL (*Structured Query Language*) : langage standard d'interrogation et de manipulation des bases de données relationnelles, permettant d'insérer, lire, modifier ou supprimer des données.

SSO (*Single Sign-On*) : mécanisme d'authentification unique permettant à un utilisateur de se connecter une seule fois pour accéder à l'ensemble des applications d'un écosystème, sans ressaisir ses identifiants.

XSS (*Cross-Site Scripting*) : catégorie d'attaque web consistant à injecter du code malveillant dans une page consultée par d'autres utilisateurs, afin d'exécuter des actions en leur nom ou de voler des données de session.

Résumé

Ce rapport rend compte d'une première année d'alternance effectuée au sein de l'équipe *IT Techno Intégration* de Limagrain, 4ème semencier mondial. Face à des opérations d'administration récurrentes dispersées entre plusieurs outils et interfaces, l'équipe a engagé la construction d'**InterOp-Box**, un portail web interne destiné à centraliser et à structurer ces interactions. Ce rapport en présente la conception, les choix techniques et les défis d'intégration rencontrés au cours de l'année.

Mots-clés : interopérabilité, portail web, architecture modulaire, **SSO**, **Microsoft Entra ID**, **Angular**, **FastAPI**.

Abstract

This report covers a first year of work-study carried out within the IT Integration & Interoperability team at Limagrain, the world's fourth-largest seed company. Faced with recurring administrative operations spread across multiple tools and interfaces, the team began building **InterOp-Box**, an internal web portal designed to centralize and streamline these interactions. This report describes its design, the technical choices made, and the integration challenges encountered over the year.

Keywords: interoperability, web portal, modular architecture, **SSO**, **Microsoft Entra ID**, **Angular**, **FastAPI**.

1 Introduction

Dans le cadre de ma première année de formation d'ingénieur en apprentissage à l'ISIMA (Université Clermont Auvergne), j'effectue mon alternance chez Limagrain, au Centre de Recherches de Chappes. J'ai intégré l'équipe *IT Techno Intégration*, rattachée au département Technologies du groupe.

Venant d'une expérience en start-up, intégrer un groupe de la taille de Limagrain change sensiblement les échelles : infrastructures plus complexes, processus plus formalisés, outils plus hétérogènes. C'est précisément ce qui m'a motivé à réaliser cette alternance, cela m'a permis de découvrir des pratiques de développement et d'intégration que je n'aurais pas rencontrées dans un contexte plus petit.

Ma mission principale cette année a porté sur le développement d'*InterOp-Box*, un portail web interne conçu pour l'équipe. Ce projet constitue le fil conducteur de ce rapport. Nous présenterons d'abord l'entreprise, l'équipe et la problématique à l'origine du projet. Nous détaillerons ensuite les choix de conception retenus pour construire une application modulaire et extensible, puis la mise en place de l'authentification unique avec Microsoft Entra ID. Enfin, à travers l'intégration de GoAnywhere MFT, nous illustrerons les difficultés que pose l'interconnexion avec un système externe aux interfaces programmatiques limitées.

2 Contexte et objectifs de l'étude

2.1 Présentation de l'entreprise

2.1.1 Origines, statut et identité

Limagrain est une coopérative agricole créée en 1965 par un groupe d'agriculteurs de la plaine de Limagne-Val d'Allier, dans le Puy-de-Dôme. Aujourd'hui, l'entreprise est toujours détenue par environ 1 300 agriculteurs adhérents. Son siège social, baptisé l'Atrium, se trouve à Saint-Beauzire (63360), à quelques kilomètres de Clermont-Ferrand.

L'entreprise possède deux sites majeurs dans le Puy-de-Dôme : l'Atrium, son siège social à Saint-Beauzire, et le Centre de Recherches de Chappes (CRC), situé à Chappes (63720). C'est sur ce dernier site que je suis basé pour mon alternance.

2.1.2 Une organisation en Business Lines

Limagrain est structuré en plusieurs branches, appelées Business Lines :

- **Limagrain Field Seeds (LFS) :** semences de grandes cultures (blé, maïs, colza, tournesol). C'est le cœur historique du groupe.
- **Limagrain Vegetable Seeds (LVS) :** semences potagères pour les professionnels du maraîchage et les jardiniers. Limagrain est numéro 1 mondial sur ce segment.
- **Jacquet Brossard :** boulangerie et pâtisserie industrielles, qui valorisent une partie des céréales produites par les adhérents.
- **Limagrain Ingrédients :** extraction et transformation de matières premières fonctionnelles pour l'agroalimentaire et la nutrition animale.
- **Vilmorin Jardin et Limagrain Coop :** jardin grand public, collecte, approvisionnement et soutien aux agriculteurs adhérents.

2.1.3 Chiffres clés (2024-2025)

Limagrain est le 4e semencier mondial. Quelques chiffres pour donner un ordre de grandeur :

- Chiffre d'affaires : 2 452 millions d'euros (partenariats inclus).
- Effectif : 9 676 collaborateurs permanents, de 92 nationalités.
- Implantation directe dans 53 pays, ventes dans près de 150 pays.
- 67 % du chiffre d'affaires réalisé en Europe.
- 37 % des effectifs en production et opérations, 22 % en recherche et innovation, 22 % en ventes et marketing.

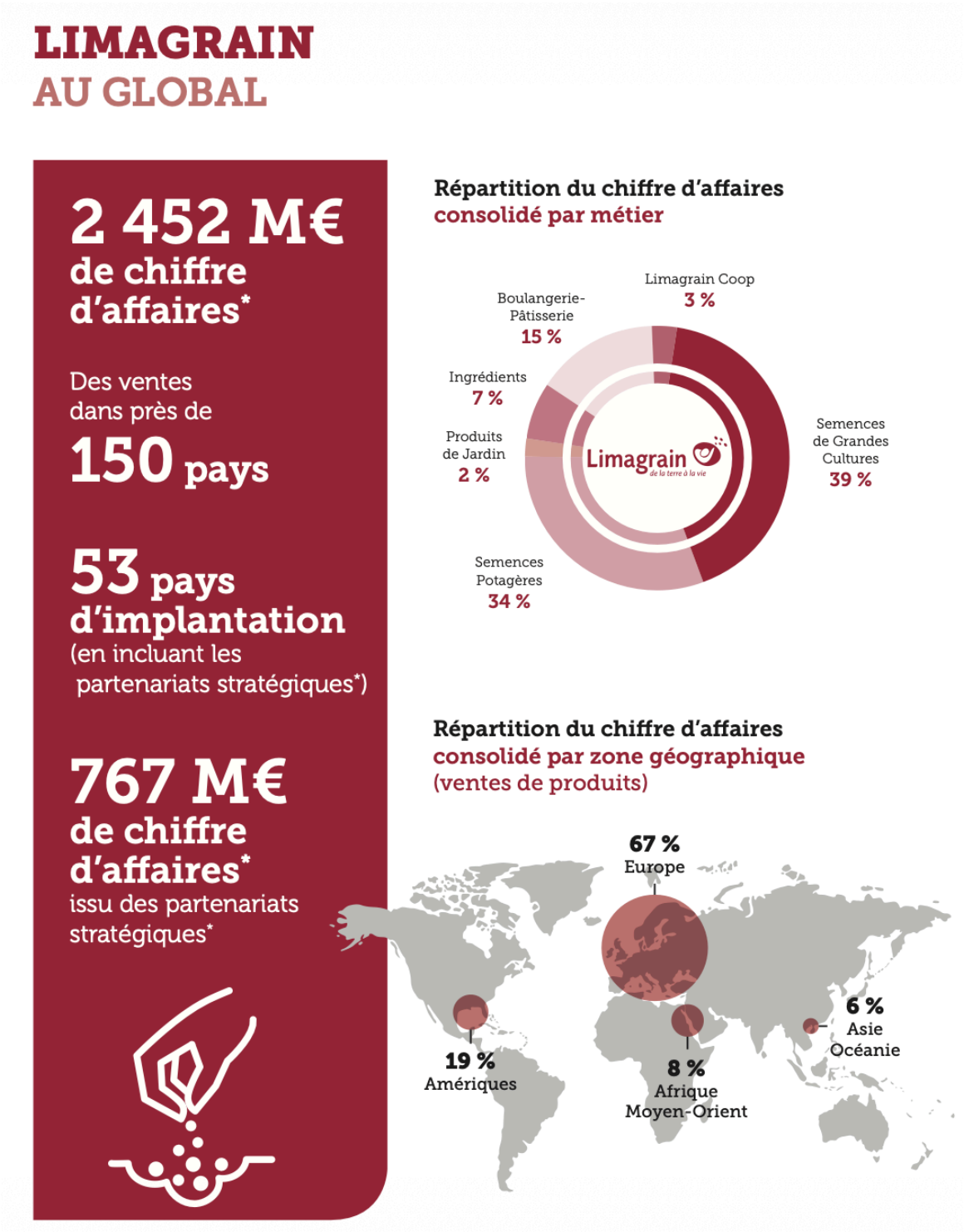


Fig. 1. – Répartition du chiffre d'affaires consolidé de Limagrain (2024)

2.1.4 Recherche et développement

La recherche est un poste important pour Limagrain : le groupe y investit environ 324 millions d'euros par an, soit à peu près 15 % de son chiffre d'affaires. Cela représente :

- Plus de 100 centres de recherche dans le monde.
- Environ 2 150 collaborateurs dédiés à la recherche et à l'amélioration des plantes.
- Un catalogue d'environ 6 000 variétés, sur 55 espèces de plantes.
- Près de 300 nouvelles variétés mises sur le marché chaque année.

Le Centre de Recherches de Chappes, où je travaille, est l'un de ces centres. Il regroupe des équipes de recherche, des laboratoires et une partie des services transverses du groupe.

2.1.5 Démarche environnementale et RSE

Limagrain a engagé une démarche de réduction de son empreinte environnementale. Les principaux objectifs affichés sont :

- Réduction de 42 % des émissions directes de CO_2 d'ici 2030 (par rapport à 2021). En 2024, la baisse atteignait déjà 10 %.
- 84 % des déchets valorisés.
- Un projet expérimental, la « Matrice de systèmes de cultures », mené sur 89 parcelles avec un budget de 7 millions d'euros, pour tester des méthodes de production plus durables.

Sur le volet social, 40,3 % des effectifs sont des femmes, et 32 % des postes de direction.

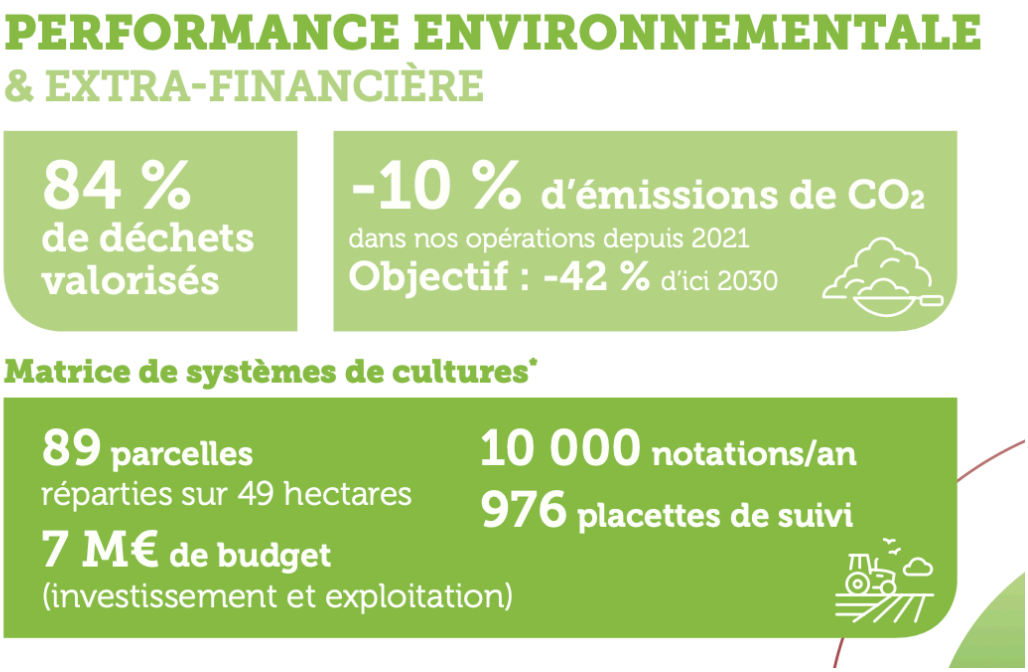


Fig. 2. – Principaux engagements environnementaux et sociaux de Limagrain (2024)

2.1.6 Équipe Intégration & Interopérabilité

Le département Technologies de Limagrain, rattaché à la Direction des Systèmes d'Information du groupe, regroupe l'ensemble des équipes informatiques transverses hébergées au Centre de Recherches de Chappes. C'est dans ce cadre que se déroule mon alternance, au sein de l'équipe *IT Techno Intégration*, dont le périmètre couvre les outils et les flux d'échange de données entre les systèmes du groupe.

L'équipe est placée sous la responsabilité de Guillaume LHOMEL (CTO), lui-même rattaché à Régis LE THIEC, DSI du groupe. Guillaume GUERIN en est le manager opérationnel et mon maître d'apprentissage. Au quotidien, l'équipe est composée de trois personnes : Guillaume GUERIN, Anass EL-GHALLAB — développeur externe spécialisé sur Talend — et moi-même.

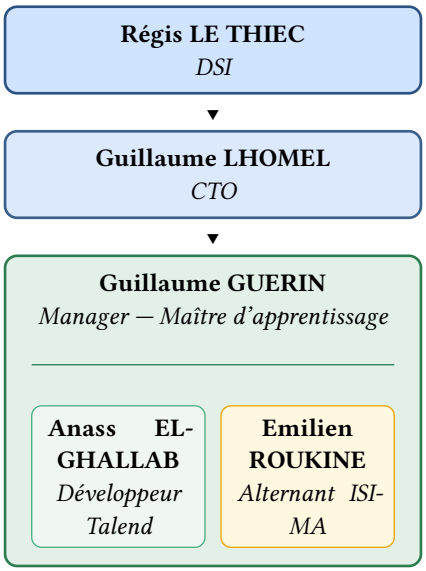


Fig. 3. – Organigramme — Position de l'équipe Intégration & Interopérabilité

Les missions au sein de l'équipe se répartissent naturellement selon les spécialités de chacun. Anass concentre son travail sur le développement des jobs Talend qui assurent les flux d'intégration entre les systèmes, son support. De son côté, Guillaume assure le pilotage des sujets stratégiques, notamment le projet de déploiement d'un API Manager pour le groupe, en plus de ses propres missions techniques. Mes responsabilités couvrent principalement le support opérationnel de niveau 2 sur GoAnywhere MFT, le développement du portail *InterOp-Box*, et la participation aux réunions et sujets transverses (tableaux de bord PowerBI, exploration de nouveaux outils, etc.).

2.2 Rappel du sujet

Parmi les différentes missions qui m'ont été confiées cette année : support GoAnywhere MFT, tableaux de bord PowerBI, le projet *InterOp-Box* s'est naturellement imposé comme sujet central de ce rapport. C'est la mission qui a mobilisé la plus grande partie de mon temps en entreprise, et c'est aussi celle dont les résultats sont les plus tangibles : une application fonctionnelle, utilisée au quotidien, avec une interface que l'on peut montrer et faire évoluer.

L'initiative est née d'un constat de Guillaume Guerin. Au fil des mois, il avait observé que de nombreuses opérations récurrentes : création de comptes utilisateurs sur des applications tierces, ouverture de tickets de support, vérification de l'état de santé des plateformes. Celles-ci reposaient encore sur des processus manuels, fragmentés entre plusieurs outils et interfaces. Ces manipulations exposaient l'équipe à des erreurs de saisie : fautes de frappe dans les identifiants, non-respect des conventions de nommage, oubli d'un paramètre de sécurité ou attribution d'un rôle inadapté. Dans le meilleur des cas, ces erreurs se traduisaient par une perte de temps pour la correction. Dans le pire, elles provoquaient des dysfonctionnements en aval, un flux configuré avec de mauvaises coordonnées, un accès accordé à un périmètre trop large, ou avec un compte utilisateur à la place d'un compte de service ou généraient des problèmes de sécurité difficiles à détecter a posteriori.

La réponse à ce constat a été de concevoir *InterOp-Box*, un portail web modulaire destiné à centraliser et à structurer les interactions de l'équipe avec ses outils. L'application ne cherche pas à remplacer les systèmes tiers ou le système de ticketing : ces plateformes restent les sources de vérité. InterOp-Box se positionne

en surcouche, comme un facilitateur et un accélérateur. Elle connecte les API de ces différents systèmes au travers d'une interface unique, guide l'utilisateur dans ses actions grâce à des formulaires pré-remplis et des contrôles de cohérence, et automatise les étapes répétitives qui n'apportent pas de valeur ajoutée humaine.

2.3 Analyse du problème

Le constat posé dans la section précédente, des opérations manuelles, des outils fragmentés, des erreurs de saisie, dessine le *pourquoi* d'InterOp-Box. Pour y répondre, trois exigences ont été formulées dès l'origine : le portail devait être *modulable*, capable d'accueillir de nouvelles fonctionnalités sans déstabiliser l'existant. Il devait être *ergonomique*, pour s'intégrer naturellement dans les routines de travail plutôt que de s'y superposer comme un outil de plus. Et il devait être *accélérateur*, sur les plans de la gestion des accès, de la qualité des tickets et de la centralisation documentaire. Reste à comprendre le *comment* : quels problèmes techniques faut-il résoudre pour transformer ces exigences en une application fonctionnelle, maintenable et déployée en production ? Trois axes se dégagent.

Le premier concerne la **modularité de l'application**. L'équipe gère des outils hétérogènes et chacun appelle des fonctionnalités propres : création de comptes, monitoring, génération de tickets structurés. Regrouper tout cela dans un monolithe rigide reviendrait à reproduire le problème de fragmentation que l'on cherche à résoudre. L'architecture technique doit permettre d'ajouter une nouvelle brique fonctionnelle de façon dynamique, sans toucher au socle existant. Il faut pouvoir délimiter clairement ce qui relève du cœur applicatif partagé (authentification, gestion de session) de ce qui relève de la logique métier propre à chaque module. C'est ce besoin qui a dicté les choix technologiques et la division de l'architecture du projet, avec notamment une découverte dynamique des modules.

Le deuxième axe porte sur l'**authentification et la sécurité d'accès**. InterOp-Box est amenée à manipuler des identifiants, des accès à des plateformes de production et des informations liées aux flux de données du groupe. L'application ne peut pas fonctionner avec un simple couple login/mot de passe local : elle doit s'intégrer au système d'identité de Limagrain, basé sur Microsoft Entra ID, afin de garantir un point d'entrée unique, une gestion centralisée des droits et une traçabilité des actions par utilisateur. La mise en place de ce SSO soulève des questions concrètes : gestion des jetons, persistance de session, protection des cookies.

Le troisième axe porte sur l'**intégration de systèmes externes**, illustrée par un module développé autour d'un outil de transfert de fichiers. À travers cet exemple, nous verrons comment un module InterOp-Box est conçu, et quelles difficultés peuvent surgir lors de ce processus. Certains outils exposent des API publiques bien documentées et facilitent l'intégration ; d'autres, au contraire, n'offrent pas ces fonctionnalités, ce qui impose de concevoir des techniques de contournement.

3 Conception d'une solution

Ce chapitre présente les choix de conception retenus pour InterOp-Box. L'objectif était de construire un portail évolutif, capable d'accueillir de nouvelles fonctionnalités sans remettre en cause l'existant, et suffisamment robuste pour être déployé en production au sein de l'infrastructure Limagrain. Les sections qui suivent détaillent d'abord le choix des technologies, puis l'organisation du code en couches distinctes, et enfin le système de modules dynamiques qui constitue le cœur architectural du projet.

3.1 Choix technologiques

Ce choix architectural conditionne les technologies retenues : chaque couche adopte le framework le plus adapté à son rôle.

3.1.1 Frontend — Angular et Tailwind CSS

Le frontend est construit avec **Angular**, un framework développé et maintenu par Google. Ce choix répond à une contrainte pragmatique : InterOp-Box est un projet né de l'initiative d'une petite équipe, sans ressources dédiées à l'assemblage et à la maintenance de bibliothèques tierces. Angular fournit nativement l'ensemble des briques nécessaires au développement d'une application web : routeur, client HTTP, injection de dépendances, système de composants réutilisables, sans qu'il soit nécessaire de dépendre de paquets externes pour ces fonctions essentielles. Cette intégration réduit la surface de maintenance et garantit la cohérence entre les différentes couches du frontend.

Le système de composants d'Angular s'inscrit directement dans la logique modulaire du projet : chaque module fonctionnel peut encapsuler ses propres vues et composants, tout en réutilisant des briques partagées pour assurer l'homogénéité de l'interface.

Le style de l'interface repose sur **Tailwind CSS**, un framework utilitaire qui permet de déclarer les règles de mise en forme directement dans le code HTML. Cette approche accélère le développement, en particulier dans un contexte full-stack où le même développeur intervient sur le backend et le frontend : elle évite les allers-retours entre fichiers de style et fichiers de template, et facilite la lecture immédiate du rendu visuel dans le code.

3.1.2 Backend — Python et FastAPI

Le backend est développé en **Python**, un langage largement répandu, polyvalent et doté d'une courbe d'apprentissage rapide. Le choix de Python est aussi un choix de cohérence au sein de l'équipe : c'est le langage principal utilisé par Guillaume Guérin, qui développe aussi sur le projet. Adopter un langage maîtrisé par le responsable du projet garantit la capacité de relecture, de revue de code et de reprise en cas d'absence.

Le framework retenu est **FastAPI**, un framework web asynchrone conçu pour la construction d'API REST performantes. Deux qualités ont motivé ce choix. La première est le support natif de l'asynchrone, qui permet de gérer efficacement les appels concurrents vers les multiples systèmes externes intégrés. La seconde est la génération automatique d'une documentation OpenAPI interactive (Swagger) : chaque endpoint de l'API est documenté et testable depuis un navigateur, ce qui constitue un outil précieux pour le développement et le débogage des intégrations.

3.2 Architecture logicielle

L'application repose sur une architecture en deux couches découplées : une interface web communique avec un serveur applicatif via une API REST, et ce dernier orchestre les appels vers les différents systèmes externes du groupe : le moteur d'intégration de données, l'outil de transfert de fichiers[1], le système de ticketing, le fournisseur d'identité (SSO Entra ID) et la base de données applicative. Les sections qui suivent détaillent l'organisation de chacune de ces couches, ainsi que le système de modules qui en constitue le cœur.

[1]GoAnywhere MFT — cet outil et les difficultés spécifiques qu'il a posées sont traités en détail dans le chapitre suivant.

3.2.1 Organisation du backend

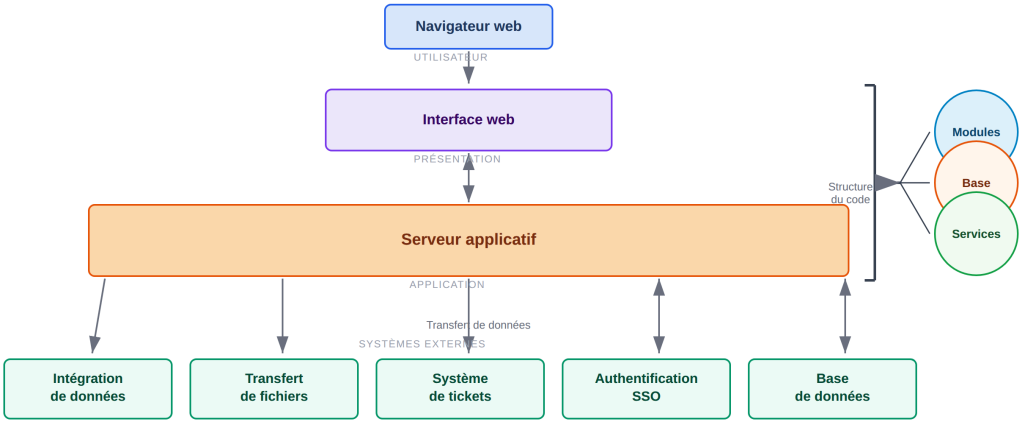


Fig. 4. – Architecture en couches d'InterOp-Box

Comme l'illustre la figure Fig. 4, le serveur applicatif est la pièce centrale de l'architecture : il reçoit les requêtes de l'interface web et orchestre les appels vers quatre systèmes externes — l'outil de transfert de fichiers (GoAnywhere), l'outil de ticketing, l'authentification SSO (Entra ID) et la base de données commune. Le code backend qui gère ces interactions s'organise en trois niveaux d'abstraction clairement délimités.

Le premier niveau regroupe les **services transverses**, partagés par l'ensemble des modules : gestion des sessions utilisateur et clients d'intégration génériques vers les systèmes externes. Ces services sont conçus comme des briques réutilisables : un module qui a besoin d'interroger l'outil de ticketing ou de lire la base de données commune consomme le service correspondant sans dupliquer la logique de connexion ni gérer lui-même les détails protocolaires.

Le deuxième niveau rassemble les **configurations de connexion** propres à chacun des quatre systèmes visibles dans la figure : points d'accès, identifiants et paramètres spécifiques à l'outil de transfert de fichiers (GoAnywhere), à l'outil de ticketing, à l'authentification SSO (Entra ID) et à la base de données commune. C'est ici que se concentre ce qui distingue un environnement de développement d'un environnement de production. Cette séparation permet d'ajuster une configuration ou d'ajouter un nouvel environnement sans toucher à la logique des services.

Le troisième niveau est celui des **modules fonctionnels**. Chaque module vit dans son propre espace, expose un routeur autonome, et ne connaît du reste de l'application que les services transverses qu'il consomme. Ce cloisonnement garantit qu'un défaut dans un module ne se propage pas aux autres, et qu'un développeur peut travailler sur un module sans maîtriser l'ensemble de la base de code.

3.2.2 Organisation du frontend

L'arborescence du frontend se divise en deux grandes parties. La première regroupe les **briques de base** de l'application, communes à tous les modules : les pages fixes (page de connexion, page d'accueil, page de rappel de mot de passe), les composants réutilisables partagés (carte de module, visionneuse de fichiers, indicateur de chargement, layout général), les services globaux (authentification, gestion des métadonnées de modules, stockage applicatif), les guards qui contrôlent l'accès aux routes selon l'identité et les droits de l'utilisateur, et les intercepteurs HTTP qui enrichissent automatiquement chaque requête sortante. La seconde partie contient les **modules fonctionnels**, chacun dans son propre répertoire, sans dépendance directe les uns envers les autres : ils consomment les services partagés, mais n'ont aucune connaissance du reste des modules.

Cette organisation présente un avantage concret pour la maintenabilité. Un développeur qui crée un nouveau module n'a pas à toucher aux pages de base, aux services globaux ni à la configuration des routes. Il dispose d'un périmètre clairement délimité, peut réutiliser tous les composants partagés, et s'appuie sur des services dont la fiabilité est garantie par le reste de l'application.

3.2.3 Système de modules dynamiques

Au lancement de l'application, le frontend interroge le backend pour récupérer la liste des modules actifs. La réponse ne contient pas uniquement des identifiants techniques : pour chaque module, le fichier de configuration déclare son nom affiché, son icône vectorielle, sa couleur, sa catégorie, sa description courte et son statut de maturité. Ce sont ces informations qui alimentent directement la page d'accueil, où chaque module apparaît sous forme d'une carte avec son titre, son icône et sa couleur (voir Annexe A). Ajouter un nouveau module revient donc à compléter ce fichier de configuration avec ses métadonnées : la page d'accueil se met à jour automatiquement au prochain chargement, sans modifier une seule ligne de code.

Le même fichier pilote la reconstruction du système de routage côté frontend. Seuls les modules présents dans la configuration et autorisés pour l'utilisateur connecté génèrent une route navigable. Cette délégation de la configuration vers un fichier externe plutôt que vers le code source simplifie considérablement le cycle de développement : il n'est pas nécessaire de recompiler l'application ni de modifier la logique applicative pour qu'un module apparaisse ou disparaisse selon l'environnement.

Ce mécanisme s'accompagne d'un système de **statuts par environnement**. Chaque module est marqué `alpha`, `beta` ou `stable`, et cette étiquette conditionne sa visibilité selon l'environnement de déploiement. Un module en cours de développement reste accessible sur les environnements de test, mais n'apparaît jamais en production avant d'être explicitement promu au statut `stable`. Cette approche élimine le risque qu'une fonctionnalité incomplète atteigne les utilisateurs finaux par inadvertance, tout en permettant de la tester en continu.

En pratique, ajouter une fonctionnalité à InterOp-Box revient à créer un répertoire côté backend et un répertoire côté frontend, puis à déclarer le module dans le fichier de configuration avec ses métadonnées et son statut. Le cœur de l'application n'a pas besoin d'être modifié : ni le point d'entrée du serveur, ni le système de routage, ni les services transverses. C'est cette propriété qui a permis à l'application de passer de quelques modules initiaux à la vingtaine actuellement en service, sans modification structurelle du cœur applicatif.

4 Authentification

InterOp-Box manipule des identifiants de connexion, des accès à des plateformes de production et des informations liées aux flux de données du groupe. La question de l'authentification ne se limite donc pas à savoir « qui se connecte » : elle conditionne la confiance que l'on peut accorder à chaque action effectuée depuis le portail. Plutôt que d'implémenter un système de comptes local, avec les risques que cela comporte (mots de passe oubliés, fuites d'identifiants, absence de politique de rotation), le choix s'est porté sur une intégration au fournisseur d'identité déjà en place chez Limagrain : Microsoft Entra ID, anciennement Azure Active Directory.

Ce choix s'inscrit dans une logique plus large. L'ensemble des outils internes de Limagrain s'appuient sur Entra ID pour l'authentification. Un collaborateur qui se connecte le matin à sa session Windows, ouvre Outlook ou accède à SharePoint utilise déjà ce même annuaire. Intégrer InterOp-Box à ce système revient à supprimer une friction d'usage : l'utilisateur n'a pas de nouveau mot de passe à créer ni à retenir, et il bénéficie automatiquement des politiques de sécurité du groupe (authentification multifacteur, expiration

de session, révocation centralisée des accès). Cette approche, connue sous le nom de *Single Sign-On* (SSO), garantit un point d'entrée unique et une traçabilité complète des connexions.

4.1 Flux d'authentification avec Microsoft Entra ID

L'authentification repose sur le protocole **OAuth 2.0 Authorization Code Flow**, un standard du web pour déléguer la vérification d'identité à un fournisseur tiers sans que l'application n'ait jamais accès au mot de passe de l'utilisateur. Le principe est le suivant : InterOp-Box ne gère pas la connexion elle-même, elle la confie à Microsoft, puis exploite le résultat.

Lorsqu'un utilisateur tente d'accéder à l'application sans être authentifié, il est redirigé vers la page de connexion d'InterOp-Box. En cliquant sur le bouton de connexion SSO, il est à son tour redirigé vers le portail de connexion Microsoft, qui prend en charge l'intégralité du processus de vérification d'identité, y compris l'authentification multifacteur si elle est activée sur son compte Limagrain.

Une fois la connexion validée par Microsoft, le navigateur est redirigé vers InterOp-Box avec un code d'autorisation temporaire. Le backend reçoit ce code, vérifie qu'il correspond bien à la demande initiale pour se prémunir contre les tentatives de falsification, puis l'échange contre un jeton d'accès en communiquant directement avec les serveurs Microsoft. Cet échange s'effectue entièrement côté serveur : le navigateur n'y participe pas, ce qui garantit que les secrets applicatifs restent confinés au backend.

Le backend utilise enfin ce jeton pour interroger l'annuaire Microsoft et récupérer le profil de l'utilisateur ainsi que ses rôles applicatifs. Ces informations sont stockées dans une session côté serveur, et un identifiant de session est renvoyé au navigateur sous forme de cookie. À partir de ce moment, l'utilisateur est authentifié et peut accéder à l'application sans avoir à présenter de nouveau ses identifiants.

Le flux complet des échanges est détaillé dans le schéma en annexe (voir Annexe B).

4.2 Gestion des sessions

Une fois l'authentification réussie, le backend crée une session côté serveur. Cette session associe un identifiant unique, généré par un générateur cryptographique, aux informations de l'utilisateur et au jeton d'accès Microsoft. L'identifiant de session est transmis au navigateur via un cookie marqué `httpOnly`, ce qui signifie qu'aucun script JavaScript exécuté dans la page ne peut y accéder. Cette propriété constitue une protection contre les attaques par injection de scripts (*Cross-Site Scripting*, XSS) : même si un attaquant parvenait à exécuter du code dans le navigateur, il ne pourrait pas lire ni exfiltrer l'identifiant de session.

À chaque requête protégée, le backend vérifie la validité de la session. Si l'identifiant est absent, inconnu ou expiré, la requête est rejetée. Ce mécanisme de vérification est implémenté sous la forme d'une dépendance FastAPI injectable, appliquée de manière déclarative sur chaque endpoint qui requiert une authentification. Concrètement, un module n'a qu'à déclarer cette dépendance dans la signature de son endpoint pour que la vérification s'effectue automatiquement, sans dupliquer la logique de contrôle.

Le choix de stocker les sessions en mémoire vive plutôt qu'en base de données a été délibéré. Avec un nombre d'utilisateurs restreint (l'équipe interne de trois personnes, et à terme quelques dizaines d'utilisateurs IT), la persistance en base aurait ajouté une complexité inutile : configuration d'un schéma, gestion des connexions, requêtes supplémentaires à chaque appel API. Le stockage en mémoire offre des temps de vérification négligeables et simplifie l'architecture. La contrepartie, la perte des sessions en cas de redémarrage du serveur, est acceptable dans ce contexte : l'utilisateur est simplement invité à se reconnecter, une opération transparente grâce au SSO.

La session expire après 24 heures. Un mécanisme de nettoyage supprime périodiquement les sessions périmées de la mémoire du serveur. La déconnexion explicite, accessible depuis l'interface, supprime immédiatement la session côté serveur et efface le cookie côté navigateur.

4.3 Rôles et permissions

L'authentification établit l'identité de l'utilisateur. Les rôles déterminent ce qu'il est autorisé à voir et à faire. InterOp-Box exploite les rôles applicatifs déclarés dans Microsoft Entra ID, qui sont intégrés directement dans le jeton d'identité émis lors de la connexion. Ce mécanisme permet de gérer les autorisations sans maintenir une base de droits séparée : l'attribution et la révocation d'un rôle se font dans l'annuaire Entra ID, et prennent effet dès la connexion suivante de l'utilisateur.

Trois rôles structurent l'accès aux modules. Le rôle `modules.admin` accorde un accès complet à l'ensemble des modules, quel que soit leur statut de maturité. Le rôle `modules.beta` ouvre l'accès aux modules en phase de validation, en plus des modules stables. Le rôle `modules.alpha` donne accès aux modules encore en développement. En l'absence de rôle spécifique, un utilisateur ne voit que les modules marqués comme stables.

Cette hiérarchie de visibilité se combine avec le système de statuts par environnement décrit dans la section précédente. Un module en `alpha` n'apparaît en production pour personne, même pour un administrateur, puisque le filtre d'environnement s'applique en amont. En revanche, sur un environnement de développement, un utilisateur disposant du rôle `modules.alpha` pourra y accéder, tandis qu'un utilisateur sans ce rôle ne le verra pas, même si le module y est techniquement déployé.

Côté frontend, deux mécanismes complémentaires assurent le respect de ces règles. Un premier garde de route (*auth guard*) vérifie, avant chaque navigation, que l'utilisateur dispose d'une session active. S'il n'est pas connecté, il est redirigé vers la page de connexion. Un second garde (*role guard*) interroge les métadonnées du module cible et confronte son statut aux rôles de l'utilisateur. Si l'accès n'est pas autorisé, la navigation est bloquée et l'utilisateur est redirigé vers la page d'accueil. Ces gardes fonctionnent de manière transparente : les modules auxquels l'utilisateur n'a pas accès n'apparaissent tout simplement pas dans la navigation, ce qui évite toute frustration liée à des boutons visibles mais inaccessibles.

Côté backend, la protection est renforcée par la vérification systématique de la session sur chaque endpoint protégé. Même si un utilisateur parvenait à deviner l'URL d'un endpoint réservé, l'absence d'une session valide ou de rôles suffisants entraînerait un rejet immédiat de la requête. Cette double barrière, frontend et backend, assure que les contrôles d'accès ne reposent pas uniquement sur la couche d'affichage, qui pourrait être contournée.

Les administrateurs disposent d'une fonctionnalité supplémentaire : un mode de vue qui leur permet de basculer entre la perspective administrateur (tous les modules visibles) et la perspective utilisateur standard (seuls les modules stables apparaissent). Ce mécanisme facilite le contrôle qualité en permettant de vérifier l'expérience réelle que percevra un utilisateur sans privilèges.

5 Intégration d'un système externe : le cas GoAnywhere MFT

GoAnywhere MFT est la plateforme de transfert de fichiers sécurisé utilisée par Limagrain pour gérer les échanges de données entre systèmes internes et partenaires externes. L'équipe Intégration & Interopérabilité en assure le support de niveau 2 et l'administration quotidienne : création de comptes utilisateurs, gestion des droits d'accès, configuration des équipes et des flux de transfert. Ce sont précisément ces opérations d'administration que les modules GoAnywhere d'InterOp-Box visent à automatiser.

Le problème est apparu dès les premières semaines de développement. GoAnywhere expose bien une API REST officielle, mais celle-ci ne couvre qu'une fraction marginale des fonctionnalités de la plateforme. Les opérations disponibles se limitent à des cas d'usage basiques (lecture de statuts, déclenchement de transferts simples) et ne permettent aucune action d'administration : ni création d'utilisateur, ni gestion d'équipe, ni modification de droits. Ces opérations ne sont accessibles que par deux moyens : l'interface web de GoAnywhere elle-même, ou la ligne de commande du serveur. Aucun des deux n'est exploitable directement depuis une API tierce. Il a donc fallu trouver des solutions de contournement pour rendre ces fonctionnalités accessibles depuis InterOp-Box.

5.1 Première tentative : rétro-ingénierie de l'API privée avec Spectral

La première piste explorée consistait à documenter l'API privée que l'interface web de GoAnywhere utilise en arrière-plan pour communiquer avec son serveur applicatif. L'idée reposait sur Spectral, un outil de linting et d'analyse de spécifications API. Le principe était le suivant : scanner l'ensemble des échanges HTTP entre le navigateur et le serveur GoAnywhere, identifier les endpoints privés sollicités par l'interface, puis utiliser un modèle d'intelligence artificielle pour générer automatiquement une documentation structurée de ces endpoints, voire un serveur MCP (Model Context Protocol) exploitable par les assistants de développement.

Les premiers tests avec un modèle local n'ont pas produit de résultats exploitables : la qualité de la documentation générée était insuffisante. Le recours à des fournisseurs d'IA externes a amélioré la qualité du rendu, mais le véritable obstacle s'est révélé structurel. GoAnywhere repose sur une architecture web ancienne, fondée sur un mécanisme de contexte de page. Chaque page de l'interface génère un jeton de contexte propre, transmis dans chaque requête envoyée au backend. Ce jeton est indissociable de la navigation réelle dans l'application : il est impossible d'en produire un valide sans avoir préalablement chargé la page correspondante dans un navigateur. Cette contrainte rend toute tentative d'appel direct aux endpoints privés inopérante, puisque chaque requête exige un contexte de navigation complet que seul un vrai parcours utilisateur peut produire.

L'exploration Spectral a donc été abandonnée après quelques jours. Le temps investi est resté limité et n'a pas eu de conséquences significatives sur le planning global, mais elle a permis de confirmer que l'approche par appel direct, qu'il s'agisse de l'API publique ou privée, était une impasse pour les besoins d'administration.

5.2 Accès direct à la base de données

La deuxième approche a été plus pragmatique. L'équipe disposait d'un accès direct à la base de données SQL Server de GoAnywhere, ce qui ouvrait la possibilité d'insérer, de modifier ou de supprimer des enregistrements sans passer par l'interface web ni par l'API. Pour certaines opérations (attribution de droits, création d'utilisateurs/groupe/projet), cette approche s'est révélée viable.

La difficulté résidait dans l'absence totale de documentation sur le schéma interne de GoAnywhere. Les noms de tables, les relations entre entités et les formats d'encodage des données n'étaient pas publics. Certaines colonnes stockaient des valeurs encodées ou sérialisées dont le format n'était pas immédiatement lisible. Un travail de rétro-ingénierie a donc été nécessaire pour comprendre la structure sous-jacente.

La méthode retenue a consisté à utiliser SQL Server Management Studio pour enregistrer une session de profilage : pendant que j'effectuais manuellement les opérations souhaitées dans l'interface web de GoAnywhere (création d'une équipe, attribution d'un droit, etc.), le profiler capturait l'intégralité des requêtes SQL exécutées par le serveur applicatif. En analysant ces traces, j'ai pu identifier les tables impliquées, l'ordre des insertions, les valeurs générées automatiquement et les contraintes d'intégrité à respecter. Pour reproduire une action depuis InterOp-Box, il suffisait ensuite de rejouer exactement la même séquence de requêtes SQL.

Cette approche fonctionne de manière fiable pour les opérations qui n'impliquent pas de logique métier côté serveur applicatif. Elle présente toutefois deux limites assumées : les actions effectuées directement en base contournent la couche métier de GoAnywhere, ce qui signifie qu'elles ne génèrent pas de logs applicatifs dans l'outil. Le couplage direct au schéma de la base crée aussi une fragilité en cas de mise à jour de GoAnywhere : un changement de structure pourrait casser silencieusement les requêtes. Pour ces raisons, l'accès direct en base a été réservé aux opérations mineures, pour lesquelles l'absence de traçabilité applicative est jugée acceptable.

5.3 Scraping comportemental : simuler un utilisateur réel

Pour la création d'un compte utilisateur, l'accès direct à la base de données s'est avéré insuffisant. L'analyse des traces SQL a montré que lors de la création d'un utilisateur via l'interface, le backend de GoAnywhere génère une clé de chiffrement propre à chaque compte. Cette clé est produite par le code applicatif du serveur, et non par la base de données : elle ne peut être ni calculée, ni copiée depuis un autre utilisateur. Les tentatives de duplication d'une clé existante ou d'insertion manuelle d'une valeur arbitraire ont systématiquement produit des comptes corrompus, incapables de se connecter ou de fonctionner correctement.

Face à cette contrainte, la seule solution viable a été de simuler intégralement le comportement d'un utilisateur humain dans l'interface web de GoAnywhere. Cette technique, connue sous le nom de *scraping comportemental* ou *web scraping*, consiste à reproduire par programme les requêtes HTTP exactes qu'émettrait un navigateur, sans en lancer un réel. Concrètement, le script interroge la page de connexion de GoAnywhere, en extrait les paramètres de session nécessaires, puis soumet les formulaires en deux étapes distinctes : d'abord le nom d'utilisateur, puis le mot de passe, chacun accompagné du jeton de contexte correspondant, extrait dynamiquement de la réponse précédente. Il navigue ensuite vers la page de gestion des utilisateurs, sélectionne le template de création approprié et remplit le formulaire avec les informations du nouveau compte. Chaque étape déclenche les requêtes HTTP normales vers le serveur applicatif, qui exécute sa logique métier complète, y compris la génération de la clé de chiffrement.

Un service dédié a été développé dans le backend d'InterOp-Box pour encapsuler cette logique. Il gère la séquence complète de connexion, la navigation vers les bonnes pages, le remplissage des formulaires avec les données fournies par l'utilisateur d'InterOp-Box, et la validation finale. Ce mécanisme de jeton de contexte par page, qui avait fait échouer l'approche Spectral, est ici contourné naturellement : puisque le script suit le parcours réel de navigation en extrayant ce jeton de chaque réponse HTTP avant de soumettre la suivante, les requêtes sont acceptées par le serveur sans nécessiter de lancer un navigateur.

Cette approche est la plus lourde techniquement des trois, mais c'est la seule qui permette d'automatiser des opérations nécessitant l'intervention de la couche métier de GoAnywhere. Elle a l'avantage de produire des comptes parfaitement fonctionnels et de générer les logs applicatifs normaux dans GoAnywhere, contrairement à l'accès direct en base. La séquence complète des appels HTTP est détaillée en annexe (voir Annexe C).

5.4 Synthèse des approches

Le développement des modules GoAnywhere illustre un cas concret d'intégration sous contrainte. L'absence d'API publique complète a imposé de combiner trois techniques distinctes selon la nature de l'opération à automatiser : l'API officielle pour les rares fonctionnalités qu'elle couvre, l'accès direct en base pour les opérations mineures ne nécessitant pas de logique serveur, et le scraping comportemental pour les opérations critiques comme la création de comptes. Chaque approche a été choisie en fonction des contraintes réelles de l'outil, sans chercher à imposer une solution uniforme là où les limites techniques ne le permettaient pas.

6 Conclusion

Cette première année d'alternance au sein de l'équipe Intégration & Interopérabilité de Limagrain a été structurée autour d'un projet central : la conception et le développement d'InterOp-Box, un portail web modulaire destiné à centraliser les interactions de l'équipe avec ses outils d'intégration. Le constat de départ était clair : des opérations récurrentes, fragmentées entre plusieurs interfaces, exposaient l'équipe à des erreurs de saisie et à une perte de temps incompatible avec les exigences de fiabilité d'un environnement de production. L'objectif était de construire une application capable de fédérer ces interactions dans une interface unique, tout en restant suffisamment souple pour accueillir de nouvelles fonctionnalités sans remettre en cause l'existant.

Les différentes missions réalisées cette année ont couvert un spectre technique large. La mise en place de l'authentification SSO avec Microsoft Entra ID m'a confronté aux mécanismes de délégation d'identité et de gestion de sessions sécurisées dans un contexte d'entreprise. Le développement de l'architecture modulaire m'a appris à concevoir un système extensible, où l'ajout d'une fonctionnalité ne nécessite pas de modifier le cœur applicatif. L'intégration de GoAnywhere MFT, avec ses contraintes d'API limitée, m'a amené à explorer des techniques de contournement variées (rétro-ingénierie de bases de données, scraping comportemental) et à faire des arbitrages pragmatiques entre la solution idéale et la solution réalisable. Au-delà du projet principal, le support opérationnel sur GoAnywhere MFT, les tableaux de bord PowerBI et la participation aux réflexions sur le déploiement d'un API Manager ont complété cette première année par des missions transverses qui m'ont permis de mieux comprendre le fonctionnement global de l'écosystème IT de Limagrain.

L'application est aujourd'hui fonctionnelle et utilisée au quotidien par l'équipe. Elle compte 23 modules répartis sur cinq catégories, dont plusieurs sont déployés en production et d'autres encore en phase de développement ou de validation. Les gains concrets sont déjà mesurables : les processus de création de comptes et de gestion des accès qui prenaient auparavant plusieurs minutes de manipulations manuelles sont désormais guidés et partiellement automatisés et les tableaux de bord offrent une visibilité sur l'activité des plateformes qui n'existait pas auparavant. Le travail n'est pas terminé : plusieurs modules restent à stabiliser, d'autres à développer, et le passage de certains statuts `alpha` vers `stable` conditionne leur disponibilité en production.

La perspective principale pour InterOp-Box est son ouverture à un périmètre plus large. L'application a été conçue pour une équipe de trois personnes, mais son architecture modulaire et son intégration au système d'identité du groupe la rendent techniquement prête pour un usage élargi à d'autres équipes IT ou métier de Limagrain. La décision de généraliser son adoption dépendra des retours d'usage et de la validation par le département. Cette année m'a permis de progresser significativement en développement full-stack, en conception d'architecture logicielle et en résolution de problèmes d'intégration. Je poursuis cette alternance avec une base technique solide et une compréhension concrète des enjeux d'interopérabilité au sein d'un groupe international.

7 Références bibliographiques

[DE LA FUENTE 2025] DE LA FUENTE Axel, *InterOp-Box — Développement d'application Web pour la recherche*, rapport d'alternance FISA 1ère année, ISIMA — Université Clermont Auvergne, soutenu le 20 juin 2025.

[MARTEL 2025] MARTEL Baptiste, *Mise en place de nouvelles plateformes d'intégration*, rapport d'alternance FISA 3ème année, ISIMA — Université Clermont Auvergne, 2025.

8 Webographie

FastAPI, *FastAPI Documentation* Ressource électronique. Tiangolo, 2024. Consulté le 20 juin 2026. Disponible sur : https://fastapi.tiangolo.com

Angular, *Angular Documentation* Ressource électronique. Google, 2024. Consulté le 20 juin 2026. Disponible sur : https://angular.dev

Microsoft, *Microsoft identity platform and OAuth 2.0 authorization code flow* Ressource électronique. Microsoft Learn, 2024. Consulté le 20 juin 2026. Disponible sur : https://learn.microsoft.com/en-us/entra/identity-platform/v2-oauth2-auth-code-flow

HelpSystems, *GoAnywhere MFT — Product Documentation* Ressource électronique. Fortra, 2024. Consulté le 20 juin 2026. Disponible sur : https://www.goanywhere.com/docs

9 Annexes

9.1 Annexe A — Page d'accueil d'InterOp-Box

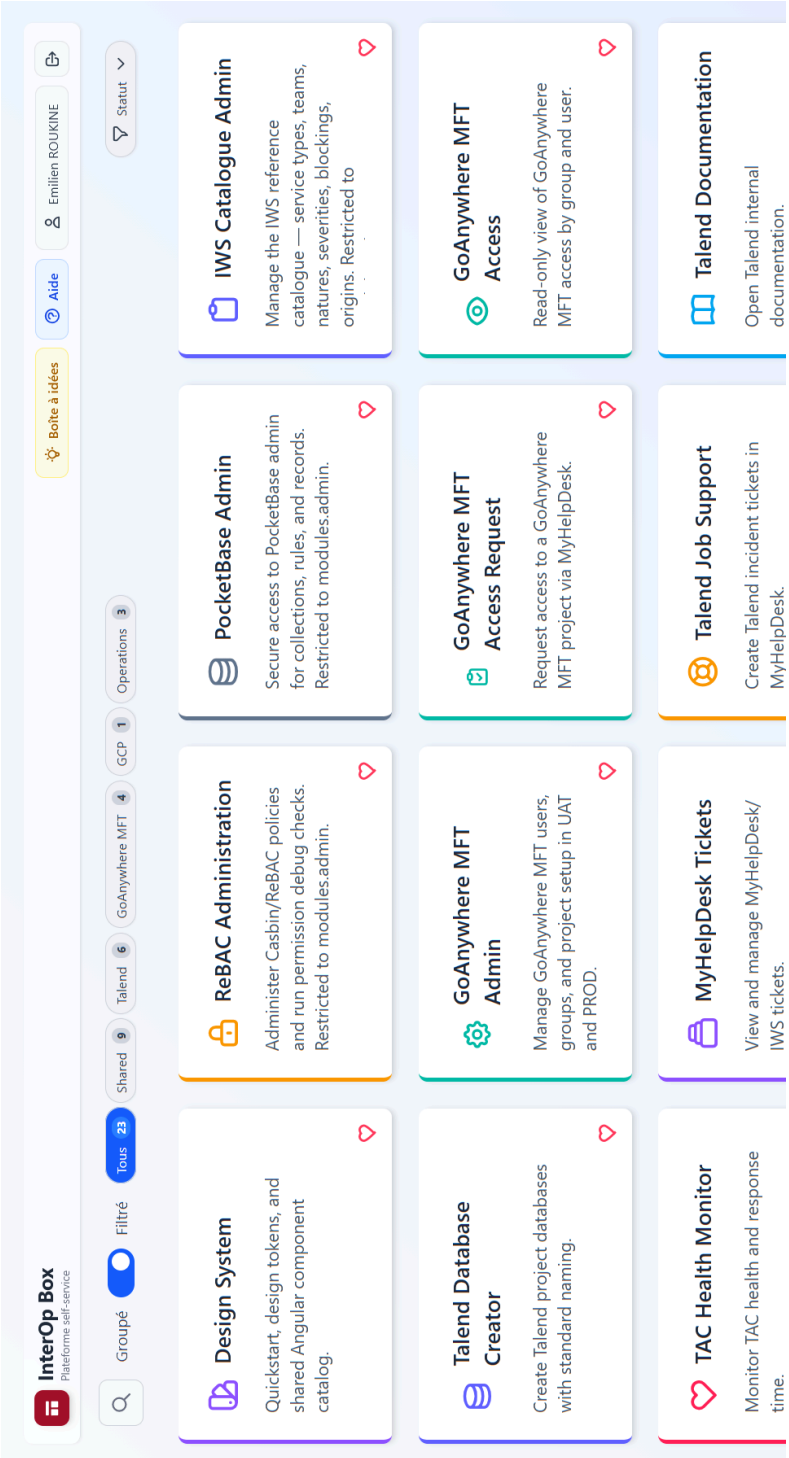


Fig. 5. – Page d'accueil d'InterOp-Box — liste des modules actifs

9.2 Annexe B — Flux d'authentification avec Microsoft Entra ID

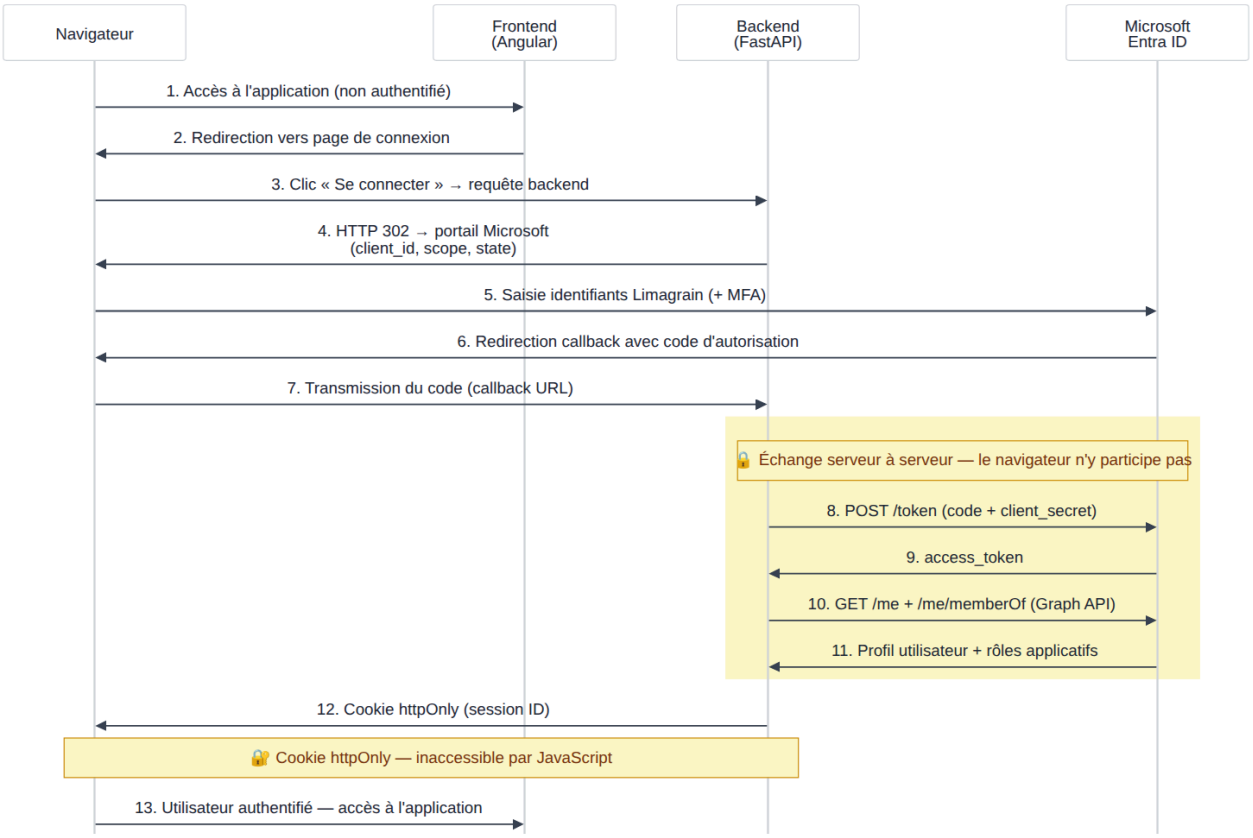


Fig. 6. – Flux d'authentification OAuth 2.0 avec Microsoft Entra ID

9.3 Annexe C — Scraping comportemental : séquence d'appels

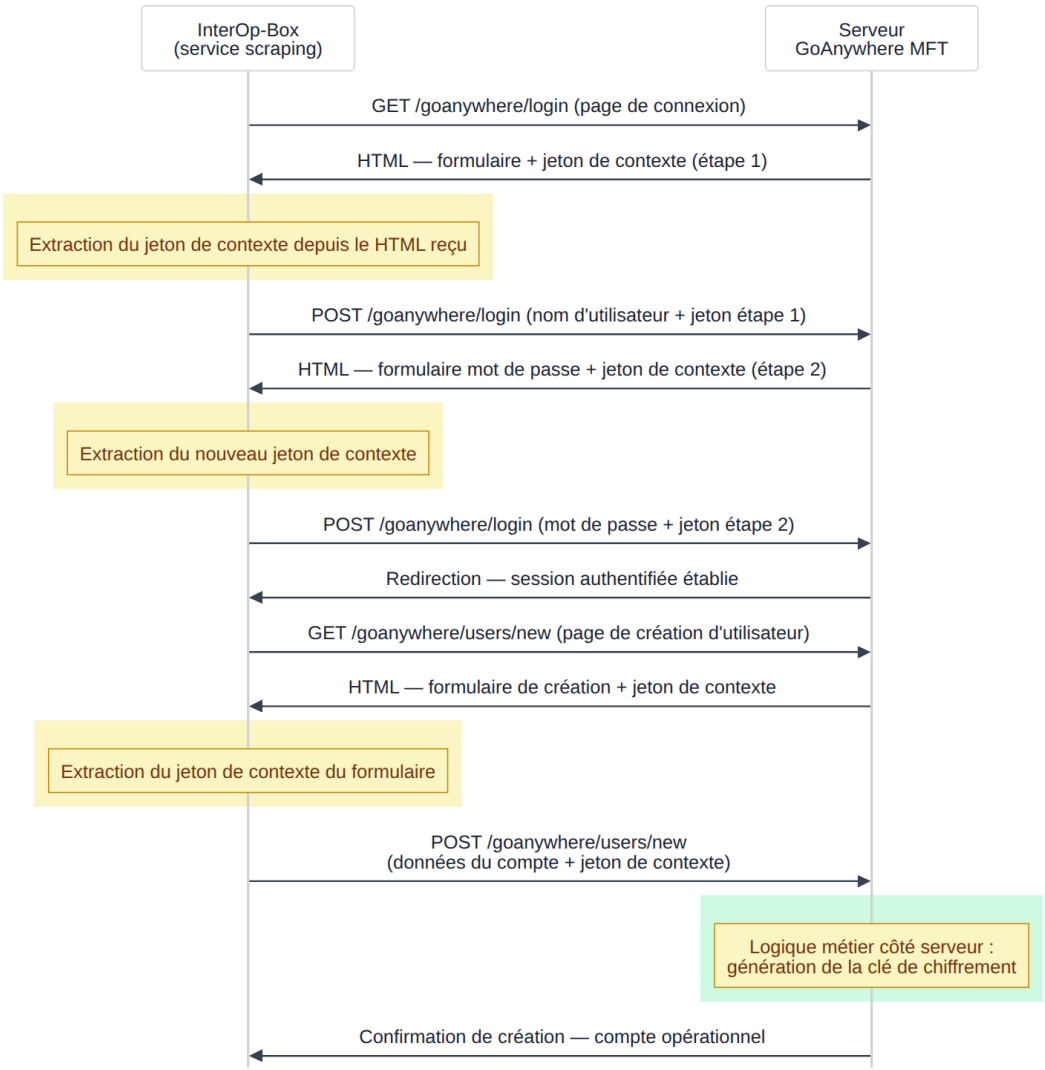


Fig. 7. – Séquence d'appels HTTP lors de la création d'un compte via scraping comportemental