DematIA

Extraction automatique d'information de documents

Dossier de soutenance

Développeur en intelligence artificielle

Ecole IA Microsoft SIMPLON

Titre professionnel n°RNCP 34757

Cédric Dromzée

Juin 2021

Table des matières

1 Introduction

1.1 Remerciements

Je remercie toute l'équipe d'Exakis-Nelite qui par sa bienveillance m'a permis de développer mes compétences dans un environnement agréable et dynamique.

Je souhaite remercier en particulier M. Edmond Brasseur, mon tuteur et Directeur technique, pour la confiance qu'il m'a accordé en m'impliquant à tous les niveaux du projet

Je remercie Guillaume Ragneau – Responsable du pôle IA qui au-delà de son pilotage depuis Nantes m'a donné l'opportunité de collaborer avec lui sur la réalisation de supports de communication.

Je remercie chaleureusement Jean-Michel Mathieu – Directeur du centre de service de Bidart et de Nantes qui en retenant ma candidature m'a permis de vivre cette expérience professionnelle.

Je tiens particulièrement à remercier mes collègues et amis Fabien Laurencet, Guillaume Cazenave et Meidi Kadri avec qui j'ai eu l'opportunité de vivre une expérience humaine exceptionnelle. Je leur suis extrêmement reconnaissant du temps et de la patience qu'ils m'ont accordé à m'aider à surmonter les obstacles et à développer mes compétences.

Je souhaite également remercier l'école IA Microsoft Simplon et en particulier Camille Dhesse et nos formateurs Louis Kuhn, Arnaud de Mouhy et Antoine Sireyjol

1.2 Exakis-Nelite

EXAKIS NELITE est une Entreprise de Services Numérique (ESN) créée en 2001 qui réalise un chiffre d'affaire de 65 millions d'euros en accompagnant des clients de toutes tailles sur leurs projets digitaux.

L'entreprise se compose de 15 agences dont 12 en France et dispose de 4 centres de services et compte plus de 500 collaborateurs

Exakis-Nelite est régulièrement labélisée partenaire de l'année par Microsoft en raison des certifications Gold sur 14 domaines d'expertise définis par Microsoft

En juin 2020, Avec Meidi Kadri, un autre alternant de l'école SIMPLON Microsoft nous avons rejoint le pôle intelligence artificielle de Biarritz pour une alternance de 12 mois dont l'objectif est de participer à la mise en œuvre du projet DematIA, une plateforme d'extraction de données issues de documents tels que des factures, contrats commerciaux, CV …

Le pôle IA présent sur les sites de Biarritz et Nantes s'est créé il y a 3 ans lors du développement de BotUI, une plateforme de Chatbot qui permet de créer des agents conversationnels à la souris et déployée sur Azure.

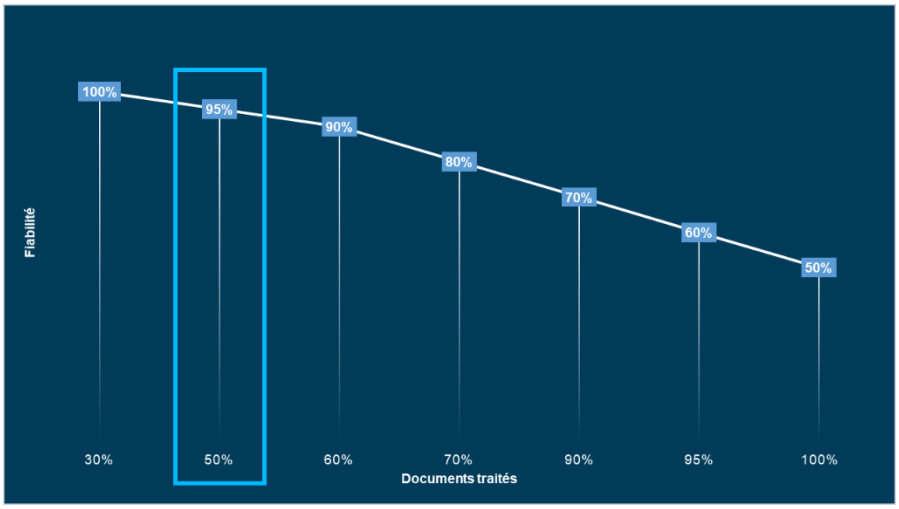
2 Analyse du besoin (C9)

Exakis-Nelite est sollicitée par ses clients pour apporter des solutions dans le cadre de la dématérialisation des documents. Les objectifs et besoins des clients sont :

- La suppression des flux papier par la digitalisation des échanges
- La valorisation des informations par l'extraction et leur utilisation
- Des prétraitements tels que l'océrisation qualitative des documents scannés …
- Des traitements automatisés incluant des tâches de formatage et de contrôle de cohérence
- L'intégration aux applications métiers : ERP, CRM, GED …
- Des gains de productivité avec la suppression de tâches fastidieuses à faible valeur ajoutée

En réponse à ces besoins, l'équipe du pôle IA a développé la plateforme Form Analyzer qui permet d'extraire les informations sur environ 50% des documents donnés en entrée avec une marge d'erreur de moins de 5% (capture d'écran en Annexe 2)

Cette extraction est réalisée par l'intermédiaire d'un algorithme qui gère de manière optimisé un pipeline de regex. Toutefois la complexité de mise en œuvre augmente si on souhaite traiter plus de documents tout en conservant une marge d'erreur acceptable ce qui génère des coûts importants.

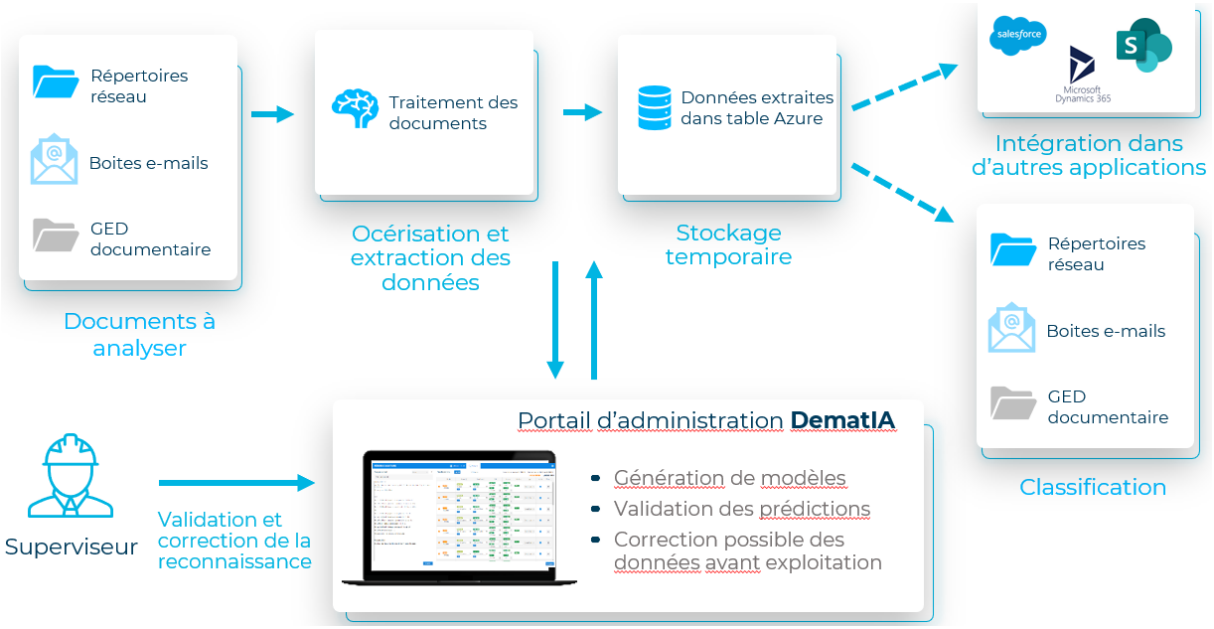


Le traitement d'une plus grande part de documents nécessite des développements coûteux pour compenser la perte de fiabilité

Les Framework de NLP sont de plus en plus utilisés pour créer des modèles statistiques sur la base de corpus texte dans l'objectif d'extraire des entités lors de prédictions.

Dans ce contexte, Exakis-Nelite souhaite évaluer si ces solutions d'intelligence artificielle pourrait améliorer ce taux de traitement sans augmenter les coûts de développement

Form Analizer + IA = Demat.IA

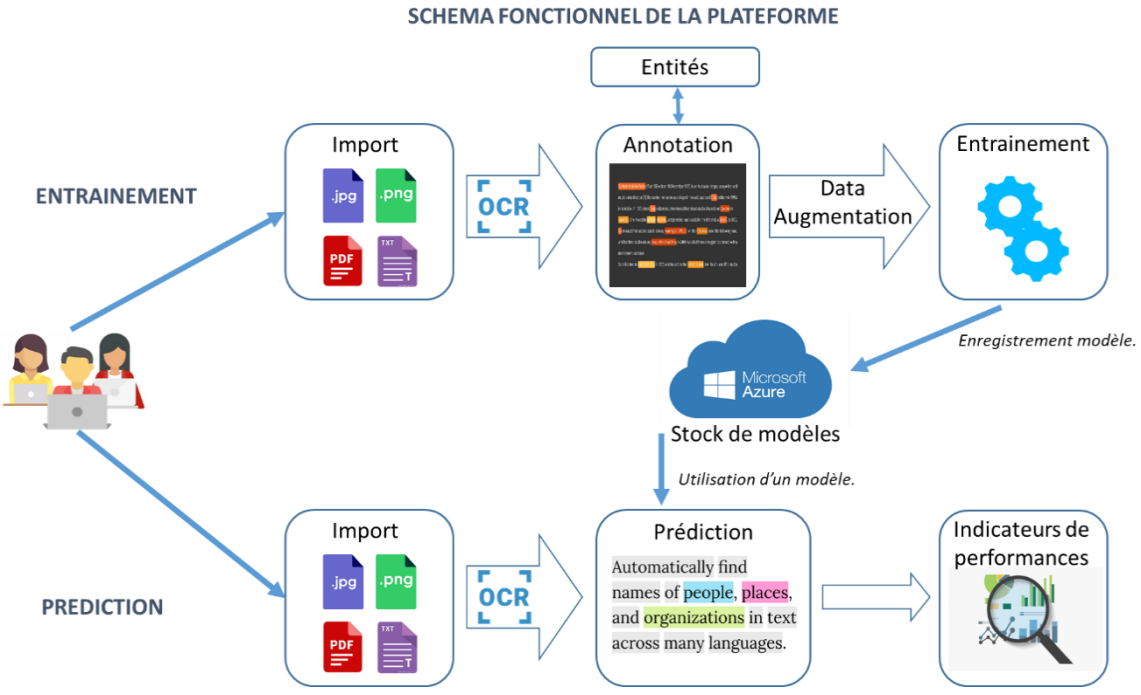


DematIA est une plateforme web prenant en entrée des documents et qui fournit en sortie des informations extraites structurées et validées pouvant être intégrées dans des outils de production

2.1 Schéma fonctionnel

A la suite d'une phase d'étude le développement de la plateforme DematIA est confirmé avec comme principales fonctions :

- D'extraire des données structurées et validées d'un document soumis par un utilisateur via une API ou une IHM en appliquant le meilleur modèle de NLP disponible
- Proposer un gestionnaire de modèles proposant des interfaces d'annotations de documents, d'entrainements de modèles, d'évaluation et de publication de modèles.



Ces fonctions doivent être accessibles depuis une interface web et utilisables par un non spécialiste de l'IA. Les utilisateurs seront des fonctionnels de services métiers tels que la comptabilité, le service de ressources humaines ou des gestionnaires clients.

Ces besoins sont synthétisés en User Stories dans le Backlog (voir Annexe 3)

Dans notre architecture nous différencions deux catégories de modèles qui seront utilisés lors des prédictions :

- Les modèles génériques
- Les modèles spécifiques

2.1.1 Les modèles génériques

Ces modèles de type NLP se composent de d'Entity Ruler et d'une partie algorithmique

Les Entity Ruler permettent d'implémenter un modèle d'expression régulière en tant qu'entité nommée personnalisée.

Par exemple, pour identifier un numéro IBAN sur des règle de 13 caractères alphanumériques ou des adresses en se liant à un dictionnaire de nom de villes.

On peut également consolider certains termes similaires dans la même catégorie par exemple Montant HT avec Total H.T.

La partie algorithmique permet de traiter des informations par rapport à leur position, en recherchant par exemple une date autour d'un libellé « émis le ».

On associe un modèle générique pour chaque type de document : il y a un modèle générique pour les factures, un autre pour les bons de commande, un autre pour des contrats …

Le modèle générique est dynamique, les dictionnaires associés sont modifiables, le modèle peut être réentraîné et peut être amélioré

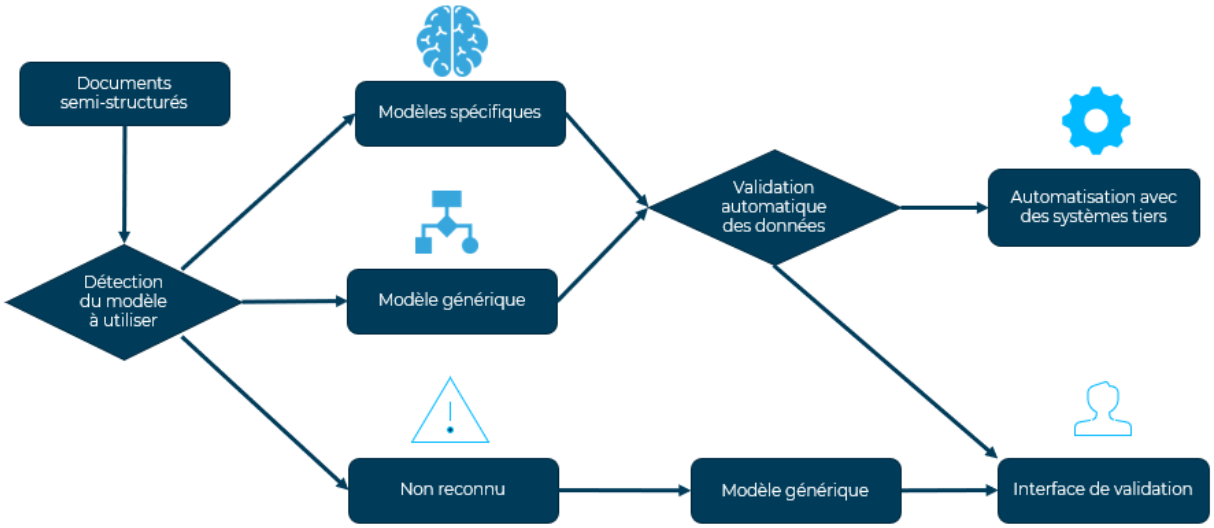
2.1.2 Les modèles spécifiques

Quand la fiabilité des prédictions sur un document est insuffisante avec le modèle générique on génère un modèle spécifique qui permet d'identifier des entités dans un document (NER Named Entity Recognition)

Ces modèles sont générés sur la base d'un ou de quelques documents qui présentent des particularités non traitables par un modèle générique. Nous avons développé un algorithme de data augmentation qui génère des instances du document initial pour entrainer un modèle statistique intéressant

C'est en permettant d'associer un modèle générique avec de multiples modèles spécifiques que nous nous arrivons à améliorer significativement de taux de traitement de documents avec une fiabilité accrue

2.1.3 Les prédictions



Lorsqu'on soumet un document pour la prédiction, un premier algorithme de classification permet de diriger le document soit :

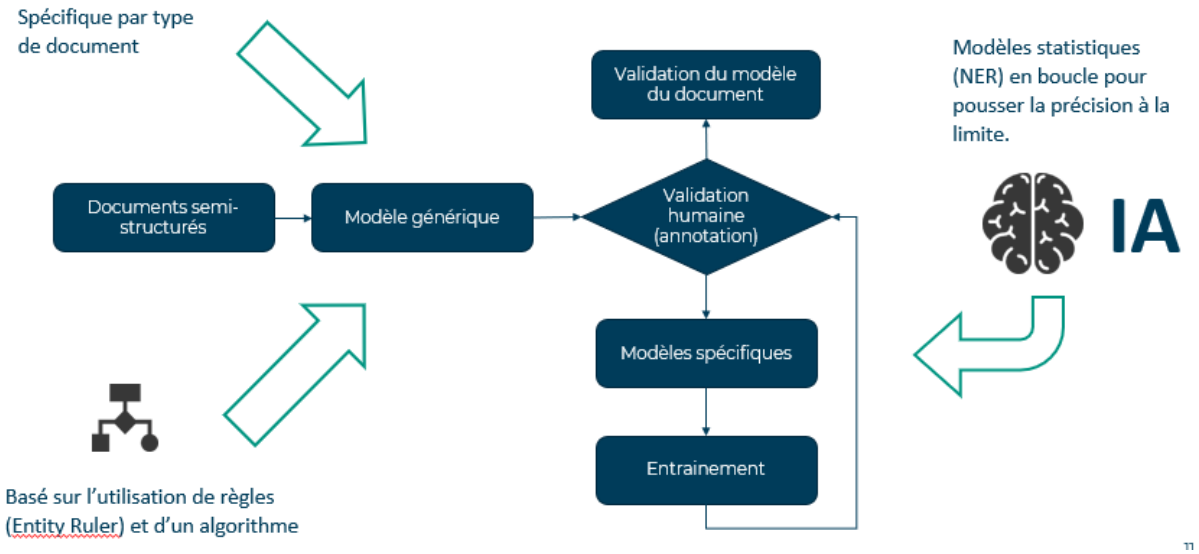
1. Vers un modèle spécifique
2. Vers un modèle générique
3. Vers une interface de traitement si le document est non reconnu

A l'issue des prédictions avec l'un des deux types de modèles, une étape de validation automatique des données permet de qualifier le traitement

- Si les scores sont bons, le workflow de traitement continu
- Dans le cas contraire le document est redirigé dans une interface de validation

Dans le cas où le classifieur ne reconnait pas le document, on applique un modèle générique et le résultat est affiché dans l'interface pour une validation humaine

2.1.4 Entrainement de modèles



Préalablement aux prédictions il est nécessaire de soumettre quelques documents types pour paramétrer le classifieur et le système de validation.

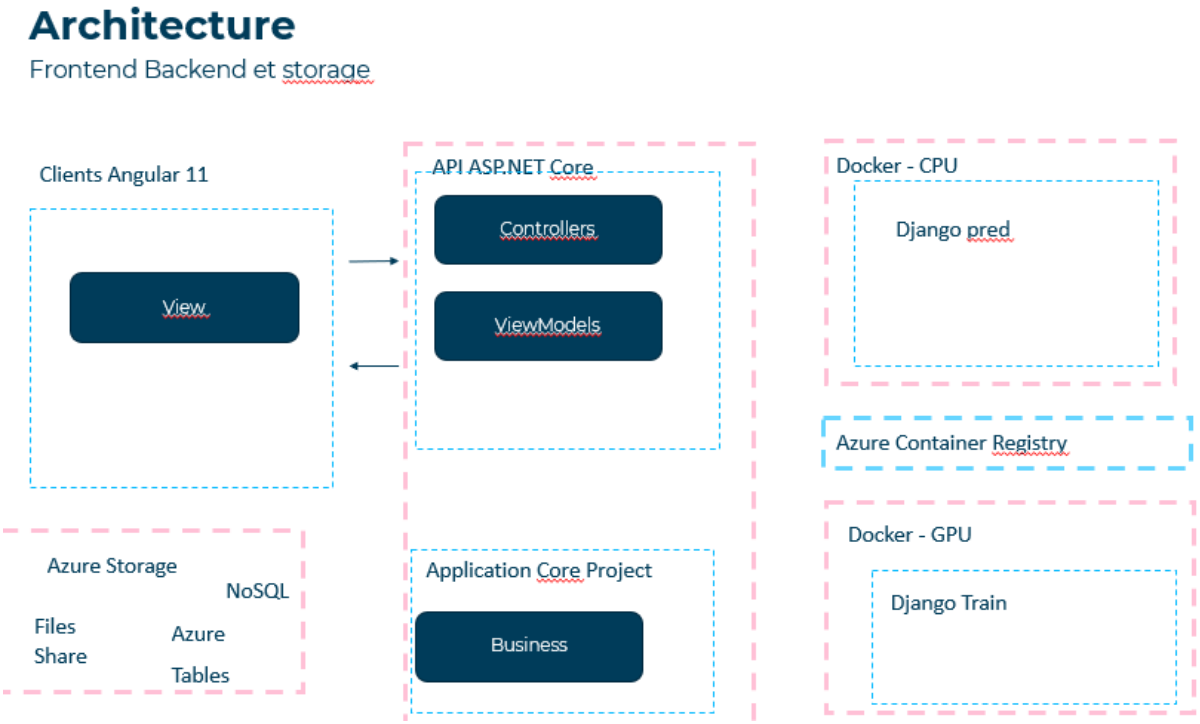
On commence par soumettre un document tel qu'une facture sur laquelle sera appliqué un modèle générique qui a été pré-entrainé sur un lot de facture.

Les entités trouvées sont affichées dans l'interface de validation.

Si l'opérateur valide que toutes les entités extraites sont conformes à ses attentes, c'est le modèle générique qui sera appliqué sur ce template de facture lors des prédictions.

Si l'opérateur identifie des corrections, il les réalise dans l'interface et un modèle spécifique sera entrainé et c'est celui-ci qui sera identifié par le classifieur pour ce template de factures lors des prédictions

2.2 Architecture (C9)



DematIA est une application Web dont le Front-end est développé à l'aide du Framework open-source Angular 11. Ce Framework très populaire initié par Google en 2009 est basé sur TypeScript

Nous l'utilisons avec la librairie de composants d'interface Kendo de Telerik

Le backend repose sur le Framework Web open-source ASP.NET Core 5.0 de Microsoft.

Il prend en charge C# et fourni de nombreuses fonctionnalités dont le développement sans compilation.

Deux API REST Django conteneurisées sont accessibles uniquement depuis l'environnement .Net permettent d'exposer les traitements proposés par le Framework de NLP Spacy

- Un conteneur déployé sur une machine virtuelle avec des ressources CPU
  Pour la data augmentation, les prédictions et les classifications
- Un conteneur déployé sur VM avec des ressources GPU
  Pour les entrainements

Les données générées par ces conteneurs tels que les fichiers JSON, les modèles générés, les métadonnées … sont stockées sur les services cloud PAAS Azure Table Storage (NoSQL) et Azure File Share

Nous utilisons la librairie .Net d'océrisation ABBYY qui permet de d'intégrer les services cloud de l'outil.

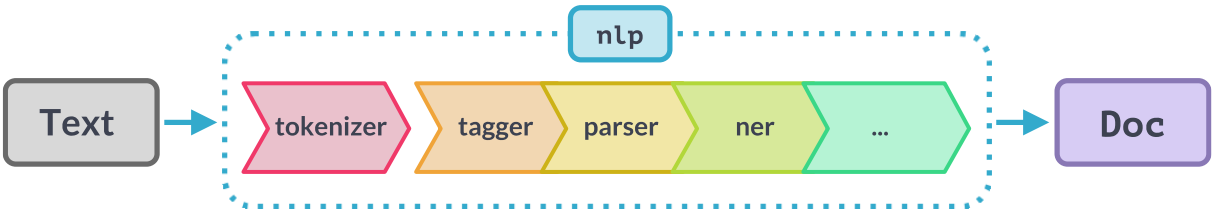
Le mot OCR (en anglais : optical character recognition) signifie reconnaissance optique de caractères ou reconnaissance de texte, une technologie qui vous permet de convertir différents types de documents tels que les documents papiers scannés, les fichiers PDF ou les photos numériques en fichiers textes modifiables et interrogeables.

2.3 Spacy

Spacy est un Framework Python de traitement automatique des langues publié sous licence MIT développé par Matt Honnibal et Ines Montani fondateurs de la société Explosion.

Le Framework est orienté production :

- Il propose de multiples modèles de réseau neuronaux statistiques préconstruits
- Il permet de former des modèles personnalisés sur leurs propres ensembles de données.
- Il propose des pipelines de traitements préconfiguré à activer



Nous avons faits plusieurs choix :

- Nous développons notre propre interface d'annotation (capture d'écran en Annexe 4)
- Nous ne découpons pas le texte en « span » (comme sur la majorité des autres outils)
- Nous n'utilisons pas les modèles préconstruits mais nous générerons nos propres modèles.
- Pour l'évaluation on split le dataset à 80 % pour le TRAIN_DATA et à 20% pour le TEST_SET

On configure un modèle vide en précisant la langue du pipeline

```
nlp = spacy.blank("fr")
```

Nous activons le « ner » dans le pipeline et nous y ajoutons les entités présentent dans le dataset TRAIN_DATA

```
if "ner" not in nlp.pipe_names:
            ner = nlp.create_pipe("ner")
            nlp.add_pipe(ner, last=True)
            # otherwise, get it so we can add labels
          else:
            ner = nlp.get_pipe("ner")

          train_data =
FileShareService.get_documents_instances(json_directories_paths=train_input.json_directory_path)

          # Ajout des labels au modèle
          for _, annotations in train_data:
            for ent in annotations.get("entities"):
              ner.add_label(ent[2])
```

Un exemple de script de train est disponible en Annexe 11

Nous sommes en cours de migration vers la v3 de Spacy sortie en 2021 et qui propose des pipelines basés sur des transformers comme BERT

Grâce aux données de test nous pouvons évaluer le modèle générer avec trois indicateurs pour chaque entité et pour le modèle global :

- Précision Détermine la proportion de vrai positif identifiés qui sont véritablement corrects.
- Rappel Proportion de vrais positifs réels identifiés correctement.
- F1 Score Moyenne harmonique entre le Rappel et la précision.

```
test_data = FileShareService.get_testing_samples(train_input.json_directory_path)

          model_scores = evaluate_NER_Model(nlp, test_data)
          model_scores_per_entity = evaluate_NER_Model_Per_Entity(nlp, test_data)
```

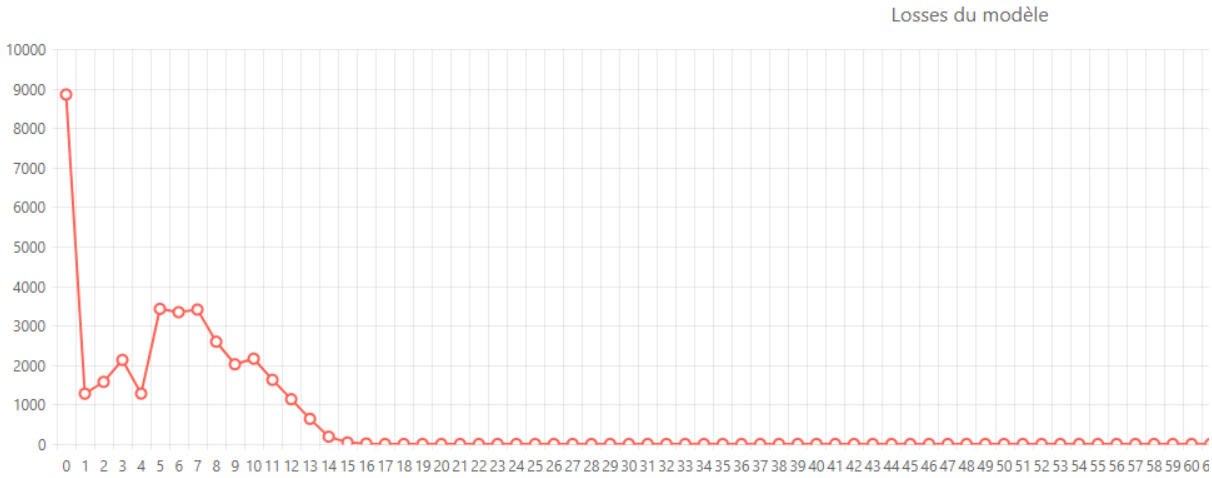
Le script de la fonction est disponible en Annexe 12

Templates | Indicateurs | Documents distincts

Entité ↑	Précision	Recall	F1
GLOBAL_MODEL	100.00	100.00	100.00
DATE_FACTURE_VALUE	100.00	100.00	100.00
IBAN_VALUE	100.00	100.00	100.00
MONTANT_HT_VALUE	100.00	100.00	100.00
MONTANT_TTC_VALUE	100.00	100.00	100.00
MONTANT_TVA_VALUE	100.00	100.00	100.00
REF_COMMANDE_VALUE	100.00	100.00	100.00
REF_FACTURE_VALUE	100.00	100.00	100.00
SIRET_VALUE	100.00	100.00	100.00
TVA_INTRACOMMUNAUTAIR...	100.00	100.00	100.00

1 | 10 items per page | 1 - 10 of 10 items

On affiche également la courbe des losses



2.4 Déploiement

Pour les clients DematIA est une application SaaS c'est-à-dire "un logiciel en tant que service" hautement disponible et accessible depuis un navigateur web.

https://demat-ia-dev.azurewebsites.net/

Mais elle est déployée sur le cloud Azure sur le modèle PAAS (Platform As A Service) c'est-à-dire que le code que nous développons est exécuté sur des plateformes mises à disposition. Ce mode nous laisse responsables du bon fonctionnement des applications.

L'architecture cloud se caractérise par :

- Service à la demande
- Accès aux ressources par le réseau
- Mise en commun des ressources
- Flexibilité et mise à l'échelle des ressources
- Service mesuré

Ce choix permet aux développeurs de se concentrer sur l'implémentation des fonctionnalités sans leur imposer la gestion de l'infrastructure matérielle

Toutefois nous avons veillé à ce que l'application puisse être déployée sur des serveurs propriétaires dits « On-Premise » que ce soit du cloud privé ou hybride ou de l'hébergement local

2.4.1 Hébergement du Backend .Net

DematIA est une **application web Azure** car elle est héberger par le service PaaS Azure Web Apps.

Une application web Azure est hébergée sur une machine Windows Server sur laquelle est installé le serveur web IIS (Internet Information Service). Microsoft est responsable de la maintenance de la machine (mise à jour et patchs de sécurité) et assure la haute disponibilité et du bon fonctionnement de cette dernière.

2.4.2 Deux VM Linux pour deux conteneurs Docker

Le déploiement se fait via la CLI Azure. Les deux images Docker trainapi et predictionapi sont pusher préalablement sur Azure Container Registry, un registre d'images Docker

- L'instance de conteneur predictionapi est déployée sur une VM CPU

- L’instance de conteneur trainapi est déployée sur VM avec GPU

2.5 Sécurité

L’application est déployée sur Azure pour réaliser des démonstrations et tester la mise en production mais sont accès n’est pas publique car nous continuons d’ajouter des fonctionnalités avant la commercialisation.

Un nom de domaine propriétaire sera configuré pour rendre accessible DematIA en https en utilisant un certificat SSL (Secure Socket Layer) afin de chiffrer toutes les requêtes

Il existe de multiples couches de sécurité.

2.5.1 Azure Active Directory

Une application Azure AD fonctionne comme une passerelle qui permet l’accès aux ressources Azure d’un abonnement en utilisant de manière sous-jacente le protocole OAuth 2.0.

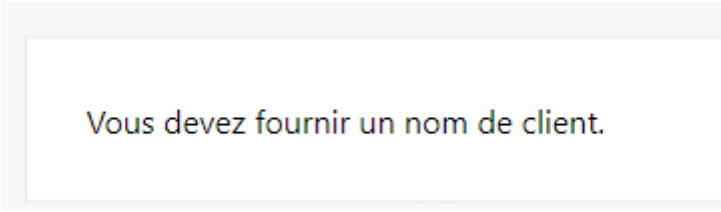
L’utilisation d’une application Azure Active Directory pour accéder aux ressources Azure d’un abonnement se fait par l’intermédiaire d’un "service principal" Azure. Un service principal représente une instance de l’application Azure AD avec un ensemble de droits, lui permettant de manipuler les ressources Azure. C’est ce service principal qui est utilisé pour s’authentifier au niveau de l’abonnement Azure.

Une application Azure AD peut définir plusieurs services principaux avec différents rôles. Un rôle définit son champ d’action possible sur l’abonnement Azure qui héberge l’application Azure AD.

Des restrictions de droits d’accès aux machines virtuelles sont configurées au niveau d’Azure Active Directory. Seul l’équipe de développement dispose de ces droits d’accès

2.5.2 Front Angular

Passé l’authentification AD nous arrivons sur la home page où il faut connaitre l’url d’un client

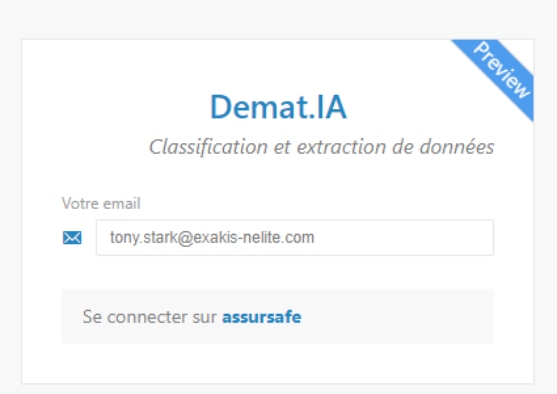


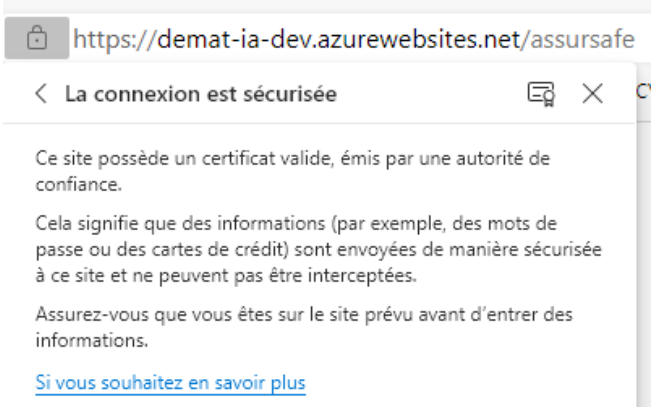
Ensuite une deuxième interface d’authentification est affichée (login)

Il n’est possible de s’authentifier qu’avec un login et mot de passe préalablement créer par un administrateur. Un utilisateur ne peut pas créer de compte (Sign Up)

La sécurisation du mot de passe :

- Mot de passe fort obligatoire (conforme CNIL)
- Hasher avec Argon2 et stockage en base sur Azure Table Storage
- Ajout d’un Salt -> chaîne générée aléatoirement et concaténée au mot de passe (donc spécifique pour chaque utilisateur)
- Et d’un “poivre” -> une chaîne globale pour tous les utilisateurs qui est stockée dans le code source applicatif (et non pas en base)
- La communication entre le Front et le back est chiffrée en TLS





L'activation du Route Guard au niveau d'Angular permet d'autoriser ou restreindre un utilisateur à naviguer sous condition vers une route. Les routes sont listées dans le tableau en annexe 11

2.5.3 Sécurité ASP.Net

Les accès et des rôles des utilisateurs sont gérés par la librairie LoggerEngine.

Au niveau des contrôleurs il est vérifié que l'utilisateur est connecté et qu'il dispose des droits pour accéder aux ressources demandées.

```
// On vérifie si l'utilisateur est connecté
if (!AuthGuard.IsLoggedIn(HttpContext.Session))
    return Unauthorized();

// On vérifie si l'utilisateur est authorisé
if (!AuthGuard.IsAuthorized(HttpContext.Session, Role.CONTRIBUTOR))
    return Forbid();
```

La chaine de connexion est encryptée en TDES, soit un triple DES (2 clés)

Par défaut, aucune donnée présente dans un compte de stockage Azure n'est accessible à un utilisateur anonyme, sauf si cela est défini de manière explicite. Par conséquent, toutes les requêtes tentant d'accéder au compte de stockage doivent être authentifiées en utilisant l'une des deux options disponibles : une clé d'accès ou signature d'accès partagé.

2.5.4 Django JWT

Les API REST Django sont sécurisés via une authentification JWT (JSON Web Token)

JWT est une chaîne JSON codée qui est passée dans les en-têtes pour authentifier les demandes. Il est généralement obtenu en hachant les données JSON avec une clé secrète.

2.6 Tests (C14)

La mise en place des tests n'est effective que pour la partie .Net Core et non sur la partie Python. L'automatisation des tests sera prochainement mis en service sur Azure DevOps

Voici le listing des tests disponibles :

Nom	Périmètre	Tests
AppStorageTests	L'Appstorage Service gère les relations avec les comptes de connexions Azure et les bases de données.	- Simule des connexions utilisateur. - Teste les connexions aux BDD. - Simule les requêtes aux BDD. - Simule les ajouts et suppressions de données.

BusinessTests	Les classes Business s'occupent des interactions entre le front et le back.	- Test de la fonction login. - Simule la création de client.
FileReaderTests	Lecture et conversions des fichiers textes.	- Test les conversions avec plusieurs paramètres, null, txt ou pdf.
OCRTests	Conversion d'un document en fichier texte.	- Simule des conversions.
PortalTests	Portal est la partie qui gère les vues de l'IHM.	- Test les connexions aux différentes vues et renvoie les code de requêtes correspondants.

Le code des tests de l'AppStorageService est en Annexe 22

Voici une liste de quelques tests fonctionnels :

Quand l'utilisateur est identifié, il est redirigé vers la plateforme
Quand l'utilisateur clique sur le bouton ajouter, le tableau passe en mode édition.
Quand l'utilisateur sélectionne un template, la vue affiche les données correspondantes.
Quand l'utilisateur clique sur le bouton Import, une fenêtre de sélection de fichier s'ouvre.
Quand l'utilisateur clique sur le bouton prédiction, le tableau des métriques du modèle s'affiche.
Quand l'utilisateur clique sur le menu, la vue correspondante est affichée de façon dynamique.
Quand l'utilisateur sélectionne un mot ou un groupe de mot dans l'outil d'annotation, un encadré entoure le span pour identifier l'entité de façon dynamique.

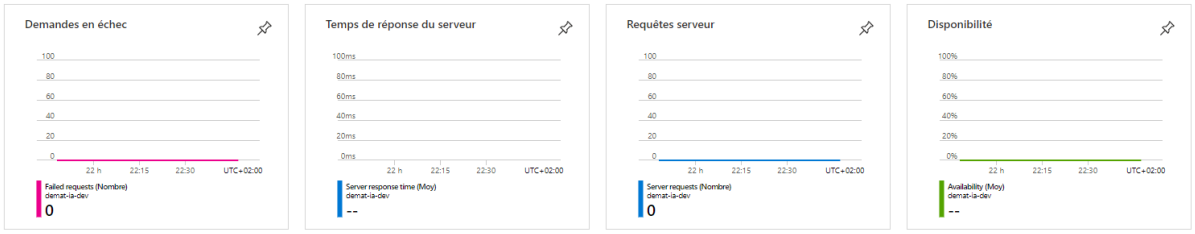
2.7 Monitoring (C15)

Azure fourni de multiples services de monitoring

2.7.1 tableau de bord

2.7.2 Insights

Application Insights est un service Azure qui permet de connaître en temps quasi réel les performances et l'utilisation de son application

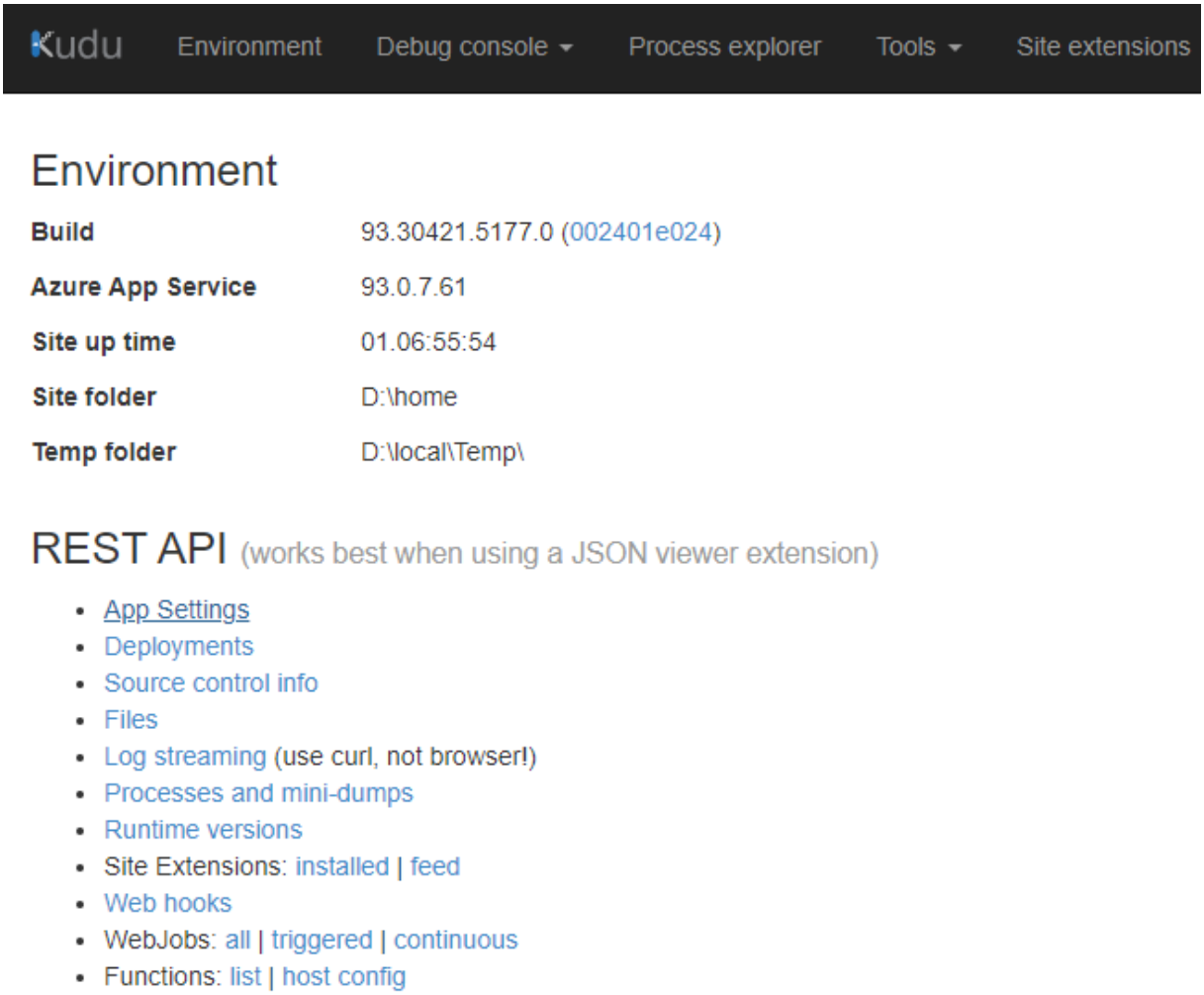


2.7.3 Kudu

Kudu est un SCM (site configuration manager) préinstallées dans le dossier ProgramFiles sur le système de fichiers du serveur web. Les applications web Azure étant des services PaaS, leurs sources ne sont pas directement accessibles.

On accède au site web d'administration KUDU via l'URL https://demat-ia-dev.scm.azurewebsites.net/

Grace au SSO (Single Sign On -> système d'authentification unique) les utilisateurs déjà connectés au portal Azure accèdent directement à Kudu sans s'authentifier. Le service utilise l'authentification Azure Active Directory basée sur OAuth2.



Parmi les fonctionnalités disponibles :

- Informations sur l'environnement
- Console d'administration

- Explorateur de processus
- Monitoring de l'exécution des traces et fonctions Azure
- API REST
- Analyser les problèmes de performance en téléchargeant le memory dump de IIS

2.7.4 Traces serveur et applicatives

Pour pouvoir détecter les erreurs applicatives pendant qu'une application web est en production (en cours d'exécution) il est nécessaire de disposer de traces d'utilisation en temps réel.

Elles sont stockées dans le répertoire LogFiles (! ces traces sont désactivées par défaut).

Il est également possible de réaliser un DaaS (Diagnostic as a Service). C'est une session de diagnostic qui audit à un moment précis l'application en collectant des informations tels que des événement Windows, sauvegarde instantanée de la mémoire, traces http

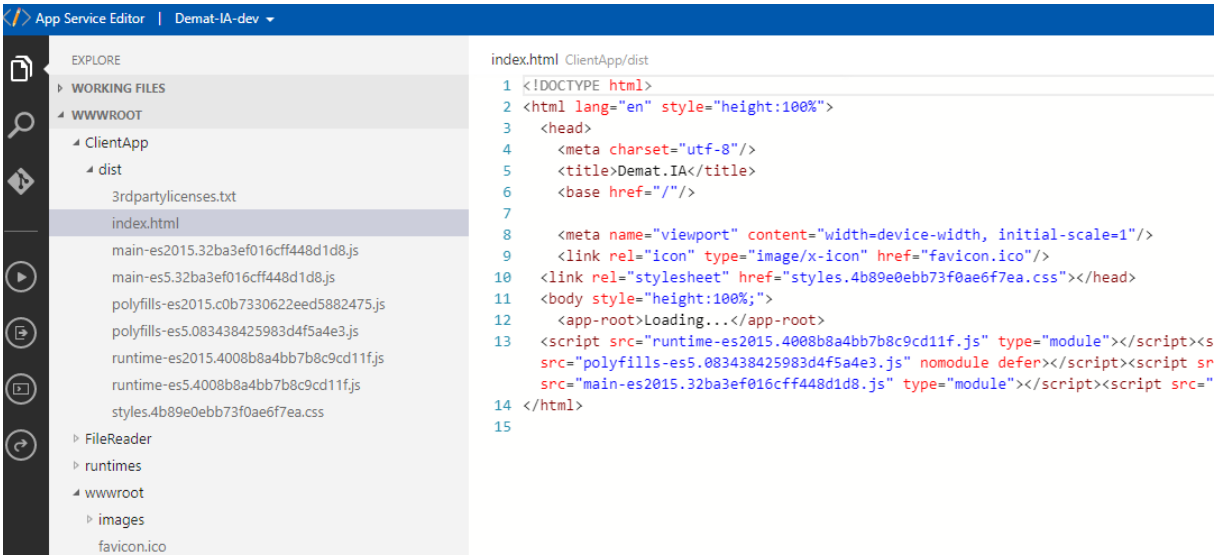
2.7.5 Monaco

Monaco est un Editeur App Service accessible par l'URL suivante :

https://demat-ia-dev.scm.azurewebsites.net/dev

L'outil permet d'éditer directement du code client (HTML, CSS, JavaScript) comme si nous étions sous Visual Studio Code.

Cet outil permet de tester rapidement du code sans à avoir à redéployer l'application



Il existe d'autres extension utiles pour le monitoring

3 Gestion de projet Agile

Dès le lancement du projet certains paramètres sont connus et définis tels que :

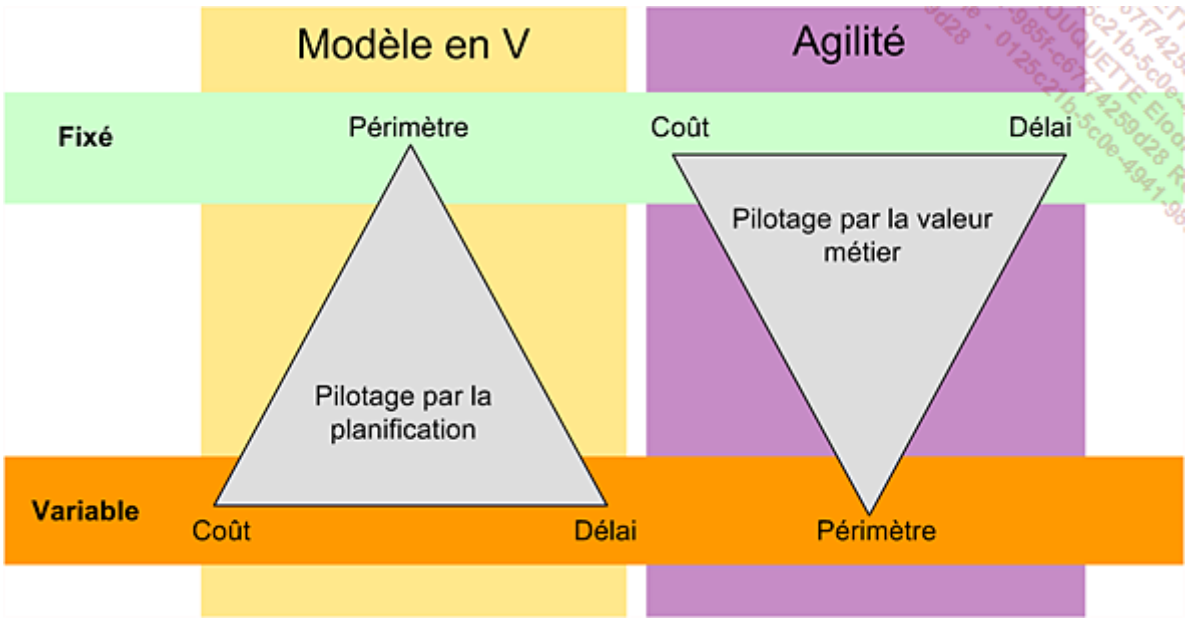
Les ressources (c'est-à-dire le coût)

- Deux alternants à temps plein durant 12 mois
- Deux développeurs expérimentés dont le nombre de jours est fixe
- Un architecte expert dont le nombre de jours est fixe

Des délais :

- Webinar devant client avec un produit fonctionnel fixé au 12 mai 2021
- Fin de CDD pour les alternants fixé au 9 juin 2021
- Nouvelles missions pour les développeurs expérimentés

Contrairement à une gestion de projet classique, les coûts et délais seront respectés mais le périmètre peut varier



Pour DematIA le périmètre s'est précisé au fur et à mesure de l'avancement du projet.

Les spécifications se sont précisées tout au long du développement et ont nécessité des adaptations continues.

Les fonctionnalités à forte valeur métier ont été développées en premier et celles ayant une faible valeur seront peut-être écartées.

La réussite du développement de ce projet est fortement liée à :

- La cohésion du groupe de travail
- L'objectif de réaliser un logiciel fonctionnel
- A la collaboration avec les clients
- A l'adaptation au changement

Il est important de retenir qu'il est impossible de s'engager sur le périmètre, mais que les coûts et délais seront respectés

3.1 Pourquoi pas Scrum

Il n'était pas possible de respecter les recommandations Scrum car :

- La durée d'un Sprint doit être fixe alors que les membres de notre équipe pouvaient être affectés à tout moment sur des missions urgentes sans en connaitre la durée pouvant réduire la capacité de production de 80 % du jour au lendemain.
- Les nécessités d'adaptation étaient trop importantes et rapides (Scrum recommande de terminer le Sprint avant d'appliquer les changements)

Toutefois nous en avons appliqué quelques principes (organisation de l'équipe …)

3.2 Kanban

Objectif de Kanban est d'éviter le gaspillage en assurant l'amélioration continue du processus

- Visualisation sous forme de tableau : voir l'état d'avancement des différentes tâches
- Gestion du flux des différentes cartes sur le tableau
- L'explicitation des normes de processus
- L'identification des opportunités d'amélioration
- La limitation du nombre de tâches en cours

3.3 L'équipe

L'équipe est un fondement essentiel de tout projet dit Agile

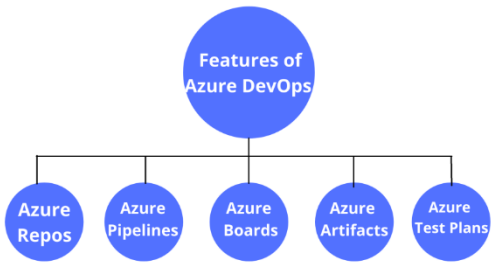
On nous a accordé une grande liberté d'organisation ce qui nous a permis de nous auto-organiser

Nous disposons d'une équipe pluridisciplinaire pour effectuer leur travail sans être dépendant de compétences ou ressources n'appartenant pas à l'équipe. Chacun dispose de multiples compétences lui permettant d'intervenir à de multiples niveaux du projet.

- Le Product Owner - Edmond
  - Responsable de la vision produit
  - Responsable du Product Backlog et de la priorisation des besoins.
  - Maximiser la valeur du produit et du travail de l'équipe
  - Définir le plan de Release
  - Accepter ou non le résultat d'un Sprint
- L'équipe de réalisation -> 4 membres
  Sa fonction principale est la transformation des User Stories contenues dans le Sprint Backlog en fonctionnalités d'un logiciel
  Chaque membre de l'équipe est donc en capacité de choisir la tâche qu'il va développer, d'en choisir (en collaboration avec les autres) l'aspect technique et bien d'autres éléments

3.4 Azure DevOps

Nous utilisons Azure DevOps qui est une boite à outils pour les équipes de développement offrant de nombreux services dans le Cloud en mode SAAS et en particulier :



- Azure Boards → outils agiles pour planifier et suivre le travail de l'équipe
- Azure Repos → dépôt Git

Nous exploiterons prochainement d'autres services :

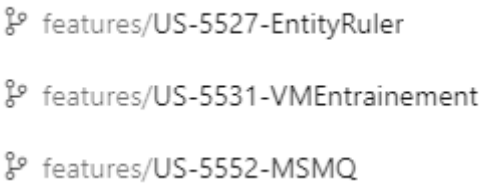
- Azure Pipelines pour l'intégration continue (CI) et la livraison continue (CD)
- Azure Test Plans

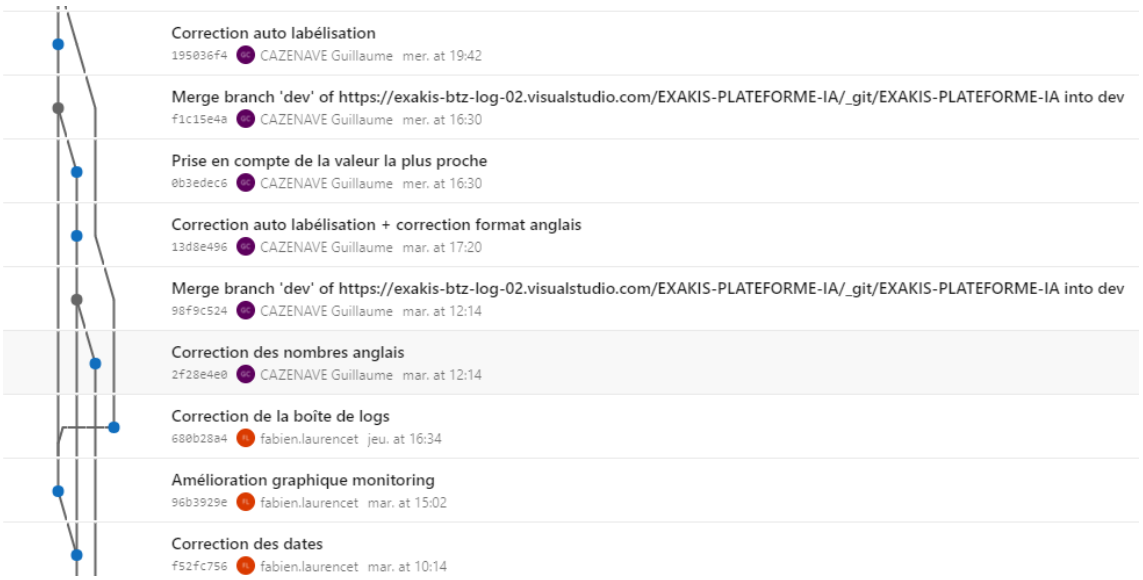
3.5 Azure Repos - Contrôle de source

Azure Repos est notre outil de contrôle de version (repository) configuré en mode distribué avec Git. Il permet à l'équipe de suivre l'évolution du code source du projet, de générer autant de branches que nécessaires et de disposer d'un mécanisme de Pull Request …

3.5.1 Workflow de branche de fonctionnalité

Chaque fonctionnalité est développée dans une branche prévue à cet effet plutôt que dans la branche main





3.5.2 historiques

L'interface Web nous permet d'accéder à l'historique de tous les Commit et de visualiser en ligne toutes les modifications de code

AppStorageService.cs -480 +9
/DEMAT.IA.FRONT/AppStorage/AppStorageService.cs

```
1   1   using Azure;
2   2   using Azure.Storage.Files.Shares;
3   3   using Azure.Storage.Files.Shares.Models;
    4 + using Azure.Storage.Queues;
    5 + using Azure.Storage.Queues.Models;
4   6   using LoggerEngine;
5   7   using Microsoft.Azure.Cosmos.Table;
6   8   using Models;
    9 + using Models.Converter;
7   10  using System;
8   11  using System.Collections.Generic;
9   12  using System.IO;
10  13  using System.Linq;
11  14  using System.Text;
12  15  using Utils;
13  16
14  17  namespace AppStorage
15  18  {
16  19      public abstract class AppStorageService
17  20      {
18      -       // <summary>
19      -       /// Nom du FileShare
20      -       /// </summary>
21      -       public static string FileShareName;
22      -
23  21          /// <summary>
24      -       /// Chaine de connexion de la base de données admin
    22 +       /// Service de TableStorage
25  23          /// </summary>
26      -       public static string AdminDatabaseConnectionString;
    24 +       public static TableStorage TableStorage { get; } = new TableStorage();
27  25
28  26          /// <summary>
29      -       /// Méthode générique de récupération des éléments de type T
    27 +       /// Service de FileShare
30  28          /// </summary>
31      -       /// <typeparam name="T"></typeparam>
```

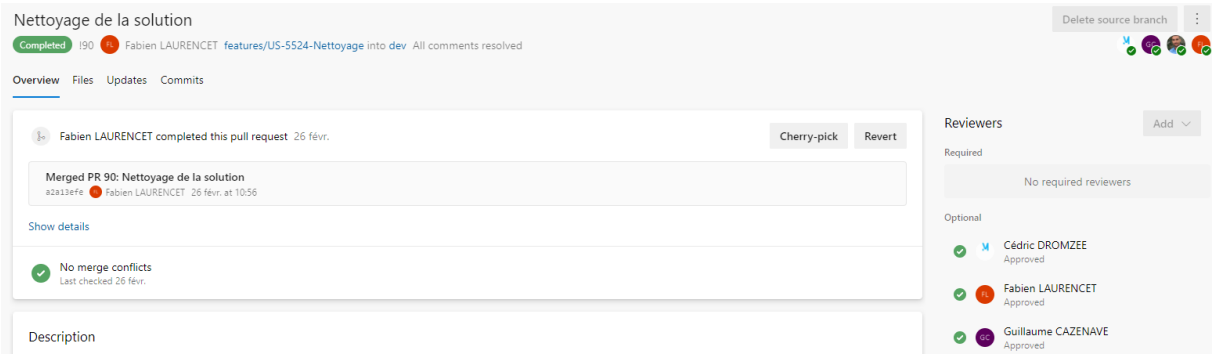
3.5.3 Pull-Request

La fonction Pull Request proposée sur l'Azure DevOps permet de disposer d'une interface Web conviviale pour :

- Discuter des changements proposés avant de les intégrer au projet officiel.
- Qu'un développeur prévienne les membres de son équipe qu'il a terminé une fonctionnalité
- Informer toutes les personnes concernées du fait qu'elles doivent réviser le code et l'intégrer à la branche main
- Si les changements posent problème, les membres de votre équipe peuvent donner du feedback

Lors d'une Pull Request, vous demandez simplement à un autre développeur (par ex., le mainteneur de projet) de faire un pull d'une branche de votre dépôt vers le sien

La PR permet également de solliciter des suggestions aux collègues



3.6 Azure Boards - planification Agile

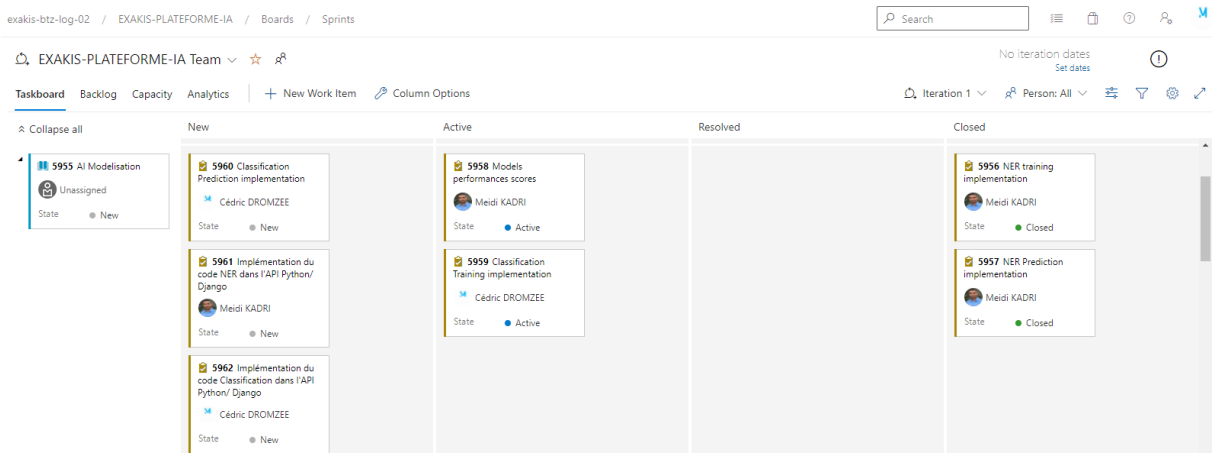
Nous disposions d'un board avec des tableaux Kanban où nos taches étaient réparties sous forme d'étiquettes ce qui nous permettait de visualiser d'un simple coup d'œil :

- L'état d'avancement du travail global et pour chaque membre
- Les étiquettes prioritaires

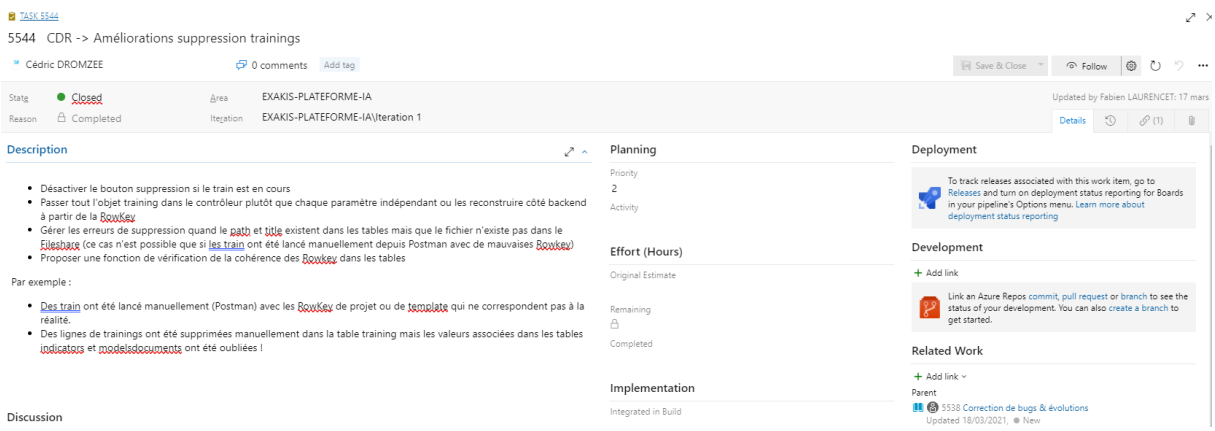
Le tableau présente des colonnes avec des étapes.

Chaque membre de l'équipe ne peut traiter qu'une étiquette à la fois

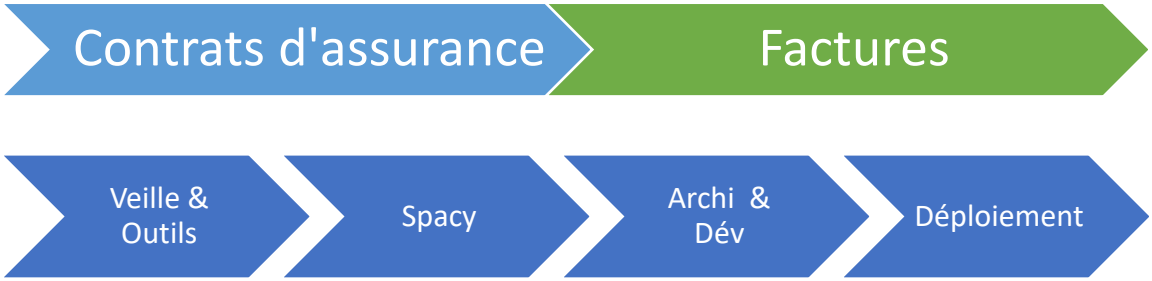
Ci-dessous l'extrait d'un Sprint



Ci-dessous un exemple de détail d'une tâche



3.7 Planification



Nous avons démarré par de la veille pour identifier tous les Framework de NLP disponibles, leurs conditions de distribution et nous les avons testé dans des notebooks.

Dès que Spacy s'est confirmé nous avons commencé à évaluer comment l'utiliser en s'appuyant sur une demande client de traitement de documents structurés : des contrats d'assurance.

Cette étape nous a permis d'identifier l'importance des annotateurs mais également de soulever la problématique du manque de documents pour l'entrainement des modèles.

Nous avons lancé le développement de la plateforme sur la base d'une première architecture mais rapidement nous avons dû nous adapter car nous n'étions pas satisfaits des résultats en soumettant des documents semi structurés tels que des factures.

Nous étions dans un processus d'amélioration continue et nous étions confrontés à de nouvelles problématiques tel que le traitement des tableaux ou de l'oubli catastrophique lors des updates des modèles.

3.8 Veille technologique

Au cours nous avons eu à réaliser de multiples veilles technologiques parfois individuellement, parfois en concurrence et parfois en groupe :

Pour chacune d'elles nous cherchons à comprendre les principes, le fonctionnement, les limites, les performances, et d'identifier les outils (Framework)

- NLP et le NER : (voir rapport E3 en Annexe 1)
- Les outils d'annotations tels que Prodigy, Doccano, UbiAI …
- Knowledge Mining
- Concurrences : autres plateformes d'extraction d'informations

Pour chacune d'elles, une période importante à plein temps était nécessaire, puis à partir des mots clés et outils identifiés la veille continuait, alimentée par des outils d'alertes, de newsletter et de curation de flux RSS (Feedly). Cette veille était synthétisée et partagée dans un notre groupe de travail sur Teams.

3.9 Communication (C18)

L'usage de la visioconférence s'est accéléré en raison des périodes de télétravail imposées en réponse à la COVID-19. Pour pallier à l'isolation notre équipe de développement se connectait chaque jour pour garder le lien social mais également pour notre Daily meeting.

Toutes les visioconférences se font avec l'application SAAS Teams de Microsoft

3.9.1 Daily

Seule l'équipe de développement participe à ce point quotidien qui a pour but de réaliser un point de synchronisation sur les tâches de développement en cours et de permettre la planification des prochaines 24 heures.

3.9.2 Point hebdomadaire du Pôle IA

Tous les jeudis cet réunion permettait de faire le point entre l'équipe de développement et le management : Direction technique, direction de pôle, directeur de site. Les développeurs présentent l'avancement et le management s'assure que le projet s'oriente toujours vers le besoin client. C'est le moment de valider des choix techniques mais également d'exposer les sollicitations de prospects.

3.9.3 Réunions d'agence et de groupe

Tous les mois il y une réunion pour tous les membres de l'agence de Bidart afin de faire le point sur les recrutements, sur tous les projets en cours

Chaque mois, tous les collaborateurs du groupe sont invités à participer à la réunion animé par le Président de Magellan afin d'être informé de tous les évènements marquant tels que les résultat financiers, l'entrée d'investisseurs, l'organisation du télétravail …

3.9.4 Comptes-rendus

Il y a eu plusieurs rapports tout au long du projet dont :

- Rapport 1 la veille sur les Framework de NLP (voir E3 en Annexe 1)
- Rapport 2 Screenshot des IHM produites
- Rapport 3 Tableau de relevé d'erreurs de prédictions

3.10 Etude de coûts

Le modèle économique de DematIA n'est pas figé et une proposition doit être réalisée pour chaque client en prenant en compte :

- L'achat de la License d'utilisation de DematIA
- L'intégration dans l'environnement de production du client
- Les futurs mises à jours de l'application
- Des prestations d'accompagnement formations, annotation, tests …

Le client doit également souscrire à des services Azure pour le déploiement de l'application ainsi que l'abonnement au service d'Océrisation ABBYY

3.10.1 Investissement humain

Exakis-Nelite est une ESN qui facture les prestations de ses collaborateurs auprès de clients. Avec 800 jours consommés et un TJM moyen de 500 € ce sont plus de 400 000 € CA qui n'ont pas été facturés.

3.10.2 ABBYY

L'outil d'OCRésition ABBYY Cloud OCR propose une grille tarifaire évoluant en fonction du nombre de pages soumises. En général c'est le client final qui s'abonne directement à ce service. Pour un traitement de 2000 pages par mois, un abonnement de 200 $ est conseillé et 0,03$ seront ajouté pour les pages supplémentaires.

https://www.abbyy.com/cloud-ocr-sdk/licensing-and-pricing/

3.10.3 ASPOSE

Nous utilisons la librairie Aspose.Total for .NET dont la licence de base avec support qui est facturée 3500 $

https://purchase.aspose.com/pricing/pdf/net

3.10.4 Kendo

La formule Kendo UI à 1 000 $ par développeur est suffisante pour disposer des composants Angular. Toutefois il est intéressant de prendre la formule DevCraft UI à 1 300 $ pour disposer des packages pour .Net

https://www.telerik.com/purchase/kendo-ui

3.10.5 Prodigy

Avant de développer notre propre outil, nous avons utilisé l'annoteur prodigy dont la licence était à 490 $ https://prodi.gy/buy

3.10.6 Visual Studio Pro

Nous utilisons plusieurs outils de développement dont Visual Studio Pro 2019 qui nécessite un abonnement Professional à 45 $ par mois https://visualstudio.microsoft.com/fr/vs/pricing/

3.10.7 AZURE

Le besoin minimum pour un environnent de production complet se compose de :

- Un plan de service P1v2 qui permet d'héberger le backend et le containeur de prédiction
- Une VM GPU à 3€ /h
- La registry pour les images docker à 12 € / mois
- Le service Azure Storage qui présente plusieurs options (coût à étudier au cas par cas)

Le plan de service Premium v2 est conçu pour fournir des performances améliorées aux applications de production et propose des machines virtuelle Dv2 à seulement 0,211 €/heure pour 1 cœur et 3,5 Go de RAM et 250 Go de stockage SSD

- Tarification App Service | Microsoft Azure
- https://docs.microsoft.com/fr-fr/azure/virtual-machines/sizes-general

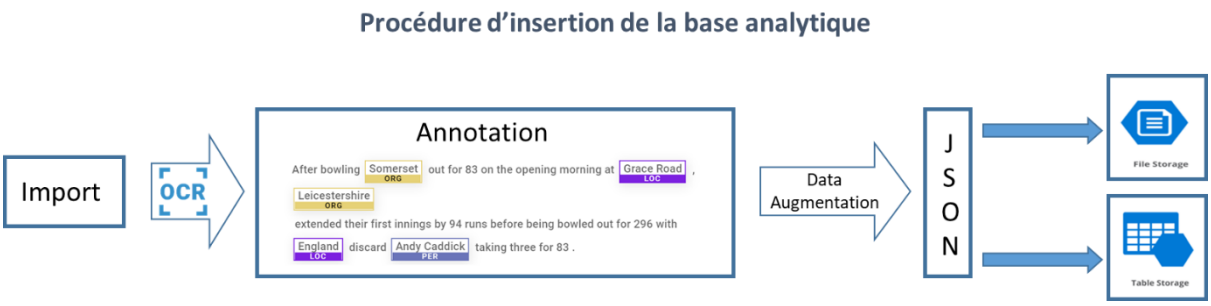
4 Les données

4.1 Processus de préparation des données

Les clients peuvent demander d'extraire des données issues de documents très variés tel que des factures, des bons de commandes, des contrats commerciaux …

La majorité de ces documents sont scannés et disponibles au format image notamment lorsque le document passe par des phases encore manuscrites tel que la signature, ou tampon par certains services.

Pour pouvoir exploiter le contenu du document nous océrisons le document via le service ABBYY qui nous renvoie le contenu au format texte.



Le processus d'insertion des données est le suivant :

- Le fichier importé est Océrisé via ABBYY
- Les entités recherchées sont labélisées dans une interface
- On génère de multiples instances de ce document via un algorithme de data augmentation
- Le fichier JSON généré est stocké sur un Fileshare d'Azure
- Les métadonnées sont enregistrées dans le service cloud table Storage azure (clés/attributs NoSQL)

4.2 Documents structurés

Nos premiers traitements se focalisent sur des contrats d'assurances qui sont des documents structurés sur lesquels nous recherchons les entités suivantes :

Labels	Description	Type
NUM_CLIENT	numéro d'identification du client.	Nombres
NUM_CONTRAT	Numéro du contrat.	Nombres
NUM_COND_GENERALES	Numéro d'identification des conditions générales utilisées pour le contrat.	Nombres
NUM_INTERCALAIRE	Numéro d'identification de l'intercalaire utilisé pour le contrat.	Nombres
ASSUREUR	Nom de l'assureur du contrat.	Texte
SOUSCRIPTEUR	Nom du client.	Texte
ADRESSE	Adresse du client.	Texte + Nombres
INDICE_CONTRAT	Indice de souscription du contrat.	Nombres flottants
DATE_DEBUT	Date d'activation du contrat.	Date
ECHEANCE	Date d'échéance	Date
DATE_BATIMENT	Date de contruction du bien assuré,	Date
SUPERFICIE	Superficie du bien assuré,	Nombres
TYPE_DOCUMENT	Est-ce un contrat, une intercalaire ou une condition générale ?	Texte
NOM_DOCUMENT	Nom du document	Texte
MONTANT	Montant de la souscription.	Nombres flottants

La principale difficulté est la très faible quantité de documents mis à disposition. De plus les documents présentent de fortes similitudes en termes de contenus mais aussi de mise en forme.

Pour entrainer un modèle efficace, il est nécessaire de fournir un volume de documents importants et variés mais nous n'avons pas trouvé de dataset pertinent pour notre cas d'usage.

Nous avons dû bien comprendre le métier du client afin d'identifier comment le contenu des documents est créé, en l'occurrence sur des conditions générales et particulières de vente. Ainsi nous avons développé un algorithme de data augmentation en mesure de générer une très grande quantité de documents avec du contenus susceptibles d'apparaitre dans un contrat.

4.3 Documents semi-structurés

L'usage le plus courant sera l'extraction de données à partir de documents semi-structurés tel que des factures, bons de commandes …

Contrairement au premier cas, nous disposons d'un nombre important et varié de documents

Les entités recherchées sont

Labels	Description	Type
DATE_FACTURE_VALUE	Date du document.	Date
REF_FACTURE_VALUE	Référence du document.	Texte + Nombres
MONTANT_HT_VALUE	Montant HT de la facture.	Nombres flottants
MONTANT_TVA_VALUE	Montant de la TVA de la facture.	Nombres flottants
MONTANT_TTC_VALUE	Montant TTC de la facture.	Nombres flottants
TVA_INTRACOMMUNAUTAIRE_VALUE	Numéro de TVA intracommunautaire de l'émetteur de la facture.	Nombres
IBAN_VALUE	Numéro IBAN de l'émetteur de la facture.	Texte + Nombres
SIRET_VALUE	Numéro de Siret de l'émetteur de la facture.	Nombres

Les données recherchées ne sont pas dans des phrases mais sont généralement sous forme d'association d'entités par exemple le Montant HT avec la valeur 2 000,00 EUR ou des entités isolées tel que le numéro de TVA intracommunautaire.

Les montants ne sont jamais les mêmes et les libellés peuvent être rédigés sous de multiples formes : ref. commande, N° de commande, CMD N° …

Dans ce contexte il est difficile d'obtenir un modèle fiable

Le processus de data augmentation nous permettra de générer de nombreuses instances de documents ce qui permettra d'entrainer le modèle avec beaucoup de variantes connues pour chaque label.

4.4 L'interface d'annotation

Pour qu'un réseau de neurone puisse entraîner un modèle NER on doit lui fournir la position de chaque entité recherchée ainsi que son étiquette.

Cette tâche est facilitée par des outils d'annotations qui affichent le document et permet à l'utilisateur d'identifier les entités qu'il recherche en lui adjoignant un label.

Au cours de la veille nous avons recherché et évalué de nombreuses interfaces d'annotations tels que Prodigy, Doccano, UbiAI …

L'outil génère un fichier au format JSON qui respecte ce formalisme :

```
TRAIN_DATA = [  {'text_A', 'entities':{  [start, end, 'LABEL'],
                                         [start, end, 'LABEL']}},
                {'text_B', 'entities':{  [start, end, 'LABEL'],
                                         [start, end, 'LABEL'],
                                         [start, end, 'LABEL']}}]
```

Par exemple

"labels": [[523, 547, "TYPE_DOCUMENT"], [642, 662, "NOM_DOCUMENT"], [761, 781, "NUM_CONTRAT"], [956, 985, "TYPE_DOCUMENT"], [998, 1026, "NUM_COND_GENERALES"], [1083, 1100, "NUM_INTERCALAIRE"],[2623, 2648, "DATE_DEBUT"], [2742, 2776, "ECHEANCE"], [3813, 3837, "TYPE_DOCUMENT"], [3914, 3934, "NOM_DOCUMENT"], [4015, 4035, "NUM_CONTRAT"], [4436, 4444, "SUPERFICIE"], [5293, 5333, "MATERIAUX"], [5399, 5414, "STANDING"], [7102, 7111, "DATE_BATIMENT"], [7203, 7216, "GARANTIE_CONDITION"], [7305, 7333, "GARANTIE"], [7406, 7426, "GARANTIE"], [7444, 7463, "GARANTIE_COEFFICIENT"], [7507, 7529, "GARANTIE"], [7816, 7840, "TYPE_DOCUMENT"], [7920, 7940, "NOM_DOCUMENT"], [8024, 8044, "NUM_CONTRAT"]]}

4.5 La data augmentation

L'algorithme de data augmentation génère des instances de document à partir d'un seul document importé dans l'objectif de fournir un dataset de qualité pour entrainer un modèle fiable.

Toutes les entités identifiées dans le fichier JSON fourni par l'annoteur seront variabilisée.

Par exemple à chaque instance créée nous pouvons :

- Modifier les montants, les dates, les numéros de factures …
- Varier l'écriture des libellés :
    - Remplacer 18/05/98 par 18 mai 1998
    - Montant HT remplacé par Total H.T. …
- Changer les noms de villes, de clients,

La data labélisation permet également de gagner du temps de labélisation manuelle. Un modèle peut être aussi efficace si il a été entrainé sur la base de 1000 factures générées à partir de 5 factures annotées que si il avait été entrainé sur la base de 1000 factures toutes annotées manuellement.

Pour variabilisé certains champs nous disposons de multiples dictionnaires d'adresses, de métiers …

4.6 Nettoyage des données

Certains traitements sont appliqués à tous les documents comme la suppression des espaces multiples ou le traitement de caractères spéciaux

```
// On supprime les espaces multiples
if (removeWithspaces)
    text = System.Text.RegularExpressions.Regex.Replace(text, @" +", " ");

// On échappe les caractères spéciaux (", \b, \f, \n, \r, \t)
if (escapeSpecialChars)
{
    text = text?.Replace("T:\\", @"");
    text = text?.Replace("\"", @"\""");
    text = text?.Replace("\b", "\\b");
    text = text?.Replace("\f", "\\f");
    text = text?.Replace("\n", "\\n");
    text = text?.Replace("\r", "\\r");
    text = text?.Replace("\t", "\\t");
}
```

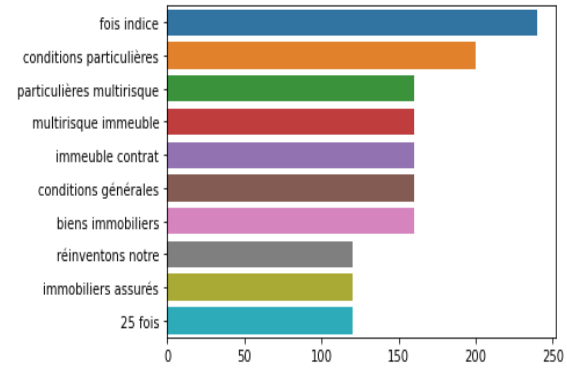
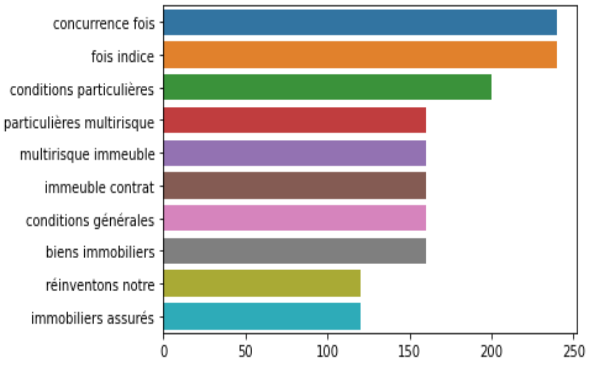
En sortie d'océrisation les documents peuvent présenter un part non négligeable de bruits qui sont généralement liés aux commentaires et signatures manuscrites, aux coups de tampons ou des sigles ajoutés par divers opérateurs.

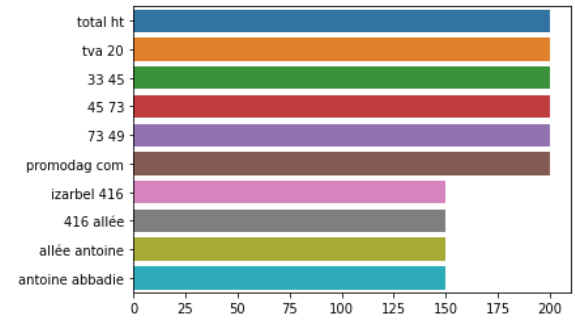
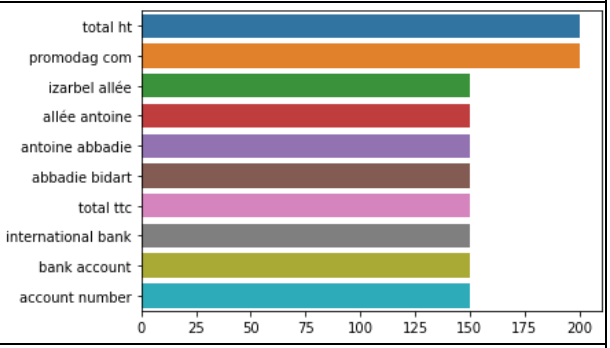
Toutefois à la suite de tests, nous préférons que le modèle s'entraine avec ces anomalies. Certaines entités seront extraites avec l'anomalie tel qu'un « * » à la place d'un « ° » que nous traiterons au cours d'une autre étape.

4.7 Exploration des données (C1)

Nous souhaitons évaluer l'impact de la présence des valeurs numériques dans nos deux dataset :
Nous classons les occurrences des 10 premiers bi-grammes sur :

- Le dataset complet dans la colonne de gauche
- Le dataset sans valeurs numériques dans la colonne de droite

<table>
<tr><th colspan="2">Lot 1 : Contrat d'assurance</th></tr>
<tr><td>Top 10 n-grams = 2 avec digits</td><td>Top 10 n-grams = 2 sans digits :</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Nuage de mots avec digits</td><td>Nuage de mots sans digits</td></tr>
<tr><td> 
</td><td> 
</td></tr>
</table>

<table>
<tr><th colspan="2">Lot 2 : Factures</th></tr>
<tr><td>Top 10 n-grams = 2 avec digits</td><td>Top 10 n-grams = 2 sans digits :</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Nuage de mots avec digits</td><td>Nuage de mots sans digits</td></tr>
<tr><td> 
</td><td> 
</td></tr>
</table>

4.8 Stockage NoSQL

Un compte de stockage Azure est une offre de service de type PaaS qui permet de stocker ses données en offrant une abstraction des contraintes de mise à l'échelle et de répartition de charge.

C'est le service Azure Table Storage qui répond à notre besoin de stockage :

- En base de données
- Données structurées
- Non relationnelles
- Données de type clé-valeur

Ce choix s'explique par de multiples arguments :

- Coût d'usage très faible (comparativement à Azure SQL)
- Flexible avec une structure sans schéma
- Support du protocole OData
- Recommandé pour de très larges volumes
- Simplicité de mise en œuvre
- Vitesse
- Vélocité pouvant être importante (Rythme d'évolution de la volumétrie de données)
- Abstraction des contraintes de mise à l'échelle et de répartition de charge

La table est accessible via une API HTTP REST sur un point de terminaison dédiée selon la nomenclature suivante :

https://[nom].[service].core.windows.net/[partition]/[objet]

https://demat-ia-dev.table.core.windows.net/[table]

Pour manipuler les différents services d'un compte de stockage Azure nous utilisons un kit de développement logiciel qui encapsule les appels aux API REST pour les applications .NET Core. Ce SDK .NET est appelé Microsoft.WindowsAzure.Storage

Une ligne représente une entité où chaque propriété est stockée dans une colonne. Chaque entité possède deux propriétés obligatoires (une clé de partition et une clé de ligne) formant un couple unique permettant son identification ainsi qu'une propriété timestamp gérée en interne par le service et qui permet de suivre la date de dernière modification.

Il est possible de gérer des entités de structures différentes dans une même table car la base est sans schéma (de type NoSQL)

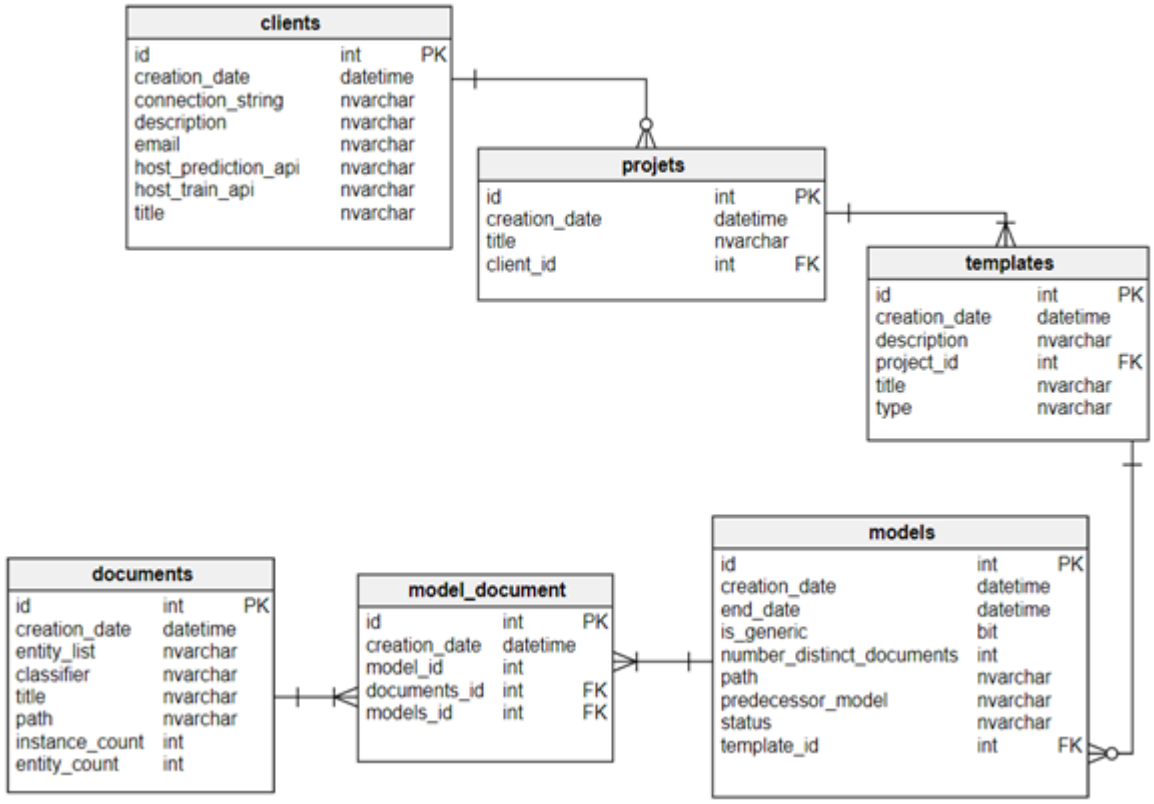
Les données sont réparties sur de multiples tables :

- Clients : Stocke les informations relatives aux clients qui utilisent la solution. Un email lors de la création du client sert de membre Administrateur.
- Membres : Contient les membres d'une organisation et leur rôle. Les rôles déterminent les autorisations et accès de certaines fonctionnalités.
- Projets : Contient la liste des projets initiés par une entreprise.
- Templates : Stocke les modèles de documents ou layout (comme un contrat, une facture…).
- Models : Stocke les informations principales d'un modèle Spacy comme la date de création, son chemin d'accès, son statut ou encore le template auquel il est rattaché.
- Documents : Contient les informations de chaque document servant à l'entrainement.

- Model_Document : Stocke les informations qui font le lien entre les modèles et les documents à partir desquels le modèle s'est entrainé.
- Models Indicator : Stocke les informations relatives aux performances d'un modèle, et notamment la précision, le recall et le F1 Score.
- Entities : Sauvegarde les caractéristiques des entités comme le nom, la couleur, le type (texte, numérique, date…), le type de reconnaissance (NER ou entity Ruler).
- Entity Rules : Contient les informations concernant les entity ruler et notamment les règles d'expressions régulières et leurs patterns pour spacy.
- Entity Variables : Stocke les informations des regex qui seront utilisées pour créer des variables lors de la génération de documents.
- Dictionaries : Contient les dictionnaires utilisés pour les regex.

4.9 SQL (C10)

Voici le MPD de la base de données



Pour les cardinalités, nous avons des relations en cascade.

- Un client peut créer plusieurs projets
- Un projet ne peut avoir qu'un seul client
- Un projet peut avoir plusieurs templates
- Un client a également n templates
- Chaque template est rattaché à un seul client

Les modèles sont également liés aux templates dont voici la cardinalité :

- 1 template pour n modèles
- 1 modèle pour 1 template.

La table model_document sert de jointure entre les documents utilisés pour les entrainements et les modèles qui sont entrainés.

- 1 document pour n model_document.
- 1 modèle pour n model_document.
- Soit N documents pour N modèles.

Nous créons une base Microsoft SQL Server local en important via l'outil Microsoft SQL Server Management Studio les scripts en annexes :

- Le script de création de la base en Annexe 19
- Le script de création des tables en **Erreur ! Source du renvoi introuvable.**
- Les script d'insertion de données en 0

Une simple requête comme ci-dessous

```
SELECT TOP (1000) [id]
    ,[creation_date]
    ,[connection_string]
    ,[description]
    ,[email]
    ,[host_prediction_api]
    ,[host_train_api]
    ,[title]
  FROM [DematIA].[dbo].[clients]
```

Permet d'extraire les 1000 premières lignes de la table clients

Results Messages

	id	creation_date	connection_string	description	email	host_prediction_api	host_train_api	title
1	1	2021-05-27 14:28:39.273	assursafeEndPointSpecimen	Client spécialisé dans les assurances	m.dupont@assursafe.fr	assursafe.predict_url	assursafe.train_url	Assursafe
2	2	2021-05-27 14:28:39.273	AxaEndPointSpecimen	Client spécialisé dans les assurances	m.dupont@axa.fr	axa.predict_url	axa.train_url	Axa
3	3	2021-05-27 14:28:39.273	MichelinEndPointSpecimen	Fournisseur automobile	m.dupont@michelin.fr	michelin.predict_url	michelin.train_url	Michelin

5 Annexes

Annexe 1. E3 – Veille

Les librairies de Reconnaissance d'Entités Nommées (REN)

The Named Entity Recognition (NER) Framework

Dans le cadre du développement d'une plateforme d'extraction automatique de données à partir de documents tels que des factures, contrats d'assurance ou questionnaires médicaux, nous recherchons une librairie de reconnaissance d'entités nommées répondant au mieux à nos contraintes.

Cet état de l'art nous aide à comprendre les bénéfices et limites de ces librairies ainsi que le contexte dans lequel elles évoluent. Ces éléments permettent la mise en œuvre d'une veille technologique garantissant le maintien des connaissances pertinentes pour ce projet.

Dans ce document les termes NER, Named Entity Recognition, REN et Reconnaissance d'entités nommées désignent le même concept et sont utilisé indifféremment.

Mots clés

NER | named-entity-recognition | Custom NER | Spacy | FlairNLP | stanford-nlp | NLP | Natural language processing | bert-language-model | pseudonymization

1 Bibliographie

Wikipedia est une encyclopédie universelle et multilingue qui permet à tous les internautes d'écrire et de modifier des articles. C'est l'encyclopédie la plus fournie et la plus consultée au monde.

- fr.wikipedia.org/wiki/Reconnaissance_d%27entit%C3%A9s_nomm%C3%A9es
- en.wikipedia.org/wiki/Named-entity_recognition

Medium est une plateforme web de blog très populaire proposant des articles techniques à jour

- medium.com/tag/named-entity-recognition
- towardsdatascience.com/tagged/named-entity-recognition
- Building a custom Named Entity Recognition model using SpaCy | Eric Landstein | 02/2020
- Named Entity Recognition and Classification with Scikit-Learn | by Susan Li | 08/2018
- Custom Entity Recognition Model using Python spaCy | by Avinash Navlani | 09/2020

GitHub est un service web d'hébergement et de gestion de développement de logiciels publiant plus de 35 millions de dépôts de projets.

- github.com/topics/ner – Liste de projets avec le mot clé « NER »
- github.com/topics/named-entity-recognition
- github.com/mariananeves/annotation-tools – Annuaire d'outils d'annotation (MAJ 12/2020)
- github.com/flairNLP FlairNLP
- https://github.com/stanfordnlp - Stanza
- github.com/JohnSnowLabs - John Snow Labs

Reddit est un site web communautaire permettant aux utilisateurs de soumettre leurs liens et de voter pour les liens proposés. Il y a beaucoup de contenu sur la programmation et la science. 20ème site web le plus populaire au monde en 2020 (6e aux États-Unis)

- reddit.com/r/LanguageTechnology
- reddit.com/r/deeplearning/

Stack Overflow et **AI Stack Exchange** sont des sites web proposant des questions et réponses sur un large choix de thèmes concernant la programmation informatique ou d'intelligence artificielle. Ils font partie du réseau de sites Stack Exchange

- Newest 'named-entity-recognition' Questions - Stack Overflow
- https://stackoverflow.com/questions/tagged/named-entity-recognition?tab=Frequent
- Newest 'named-entity-recognition' Questions - Artificial Intelligence Stack Exchange

Youtube est un site web d'hébergement de vidéos

- youtube.com/channel/UCFduT4kW_eLDbEW6XoA5F0A Chaîne de L'éditeur de Spacy et Prodygy
- youtube.com/channel/UCmFOjlpYEhxf_wJUDuz6xxQ Chaîne de John Snow Labs
- https://www.youtube.com/watch?v=9Hkr5GdbEXQ 2018 Live-Coding: Named Entity Recognition
- https://www.youtube.com/watch?v=6-fPy5j-Uzg&feature=youtu.be

Udemy est un site internet de formation en ligne qui propose plus de 57 000 cours en 65 langues

- Data Augmentation in NLP 12/2020 par Prathamesh Dahal

Twitter est un réseau social de microblogage

- https://twitter.com/hashtag/NLP
- https://twitter.com/spacy_io

Ines Montani – Influenceuse, co-fondatrice de Spacy

- https://github.com/ines
- ines.io
- https://twitter.com/_inesmontani
- User Ines Montani - Stack Overflow

Explosion – Editeur de Spacy et Prodigy

- explosion.ai
- spacy.io/usage/training#ner

Google Colab ou Colaboratory est un service cloud basé sur Jupyter Notebook et destiné à la formation et à la recherche dans l'apprentissage automatique.

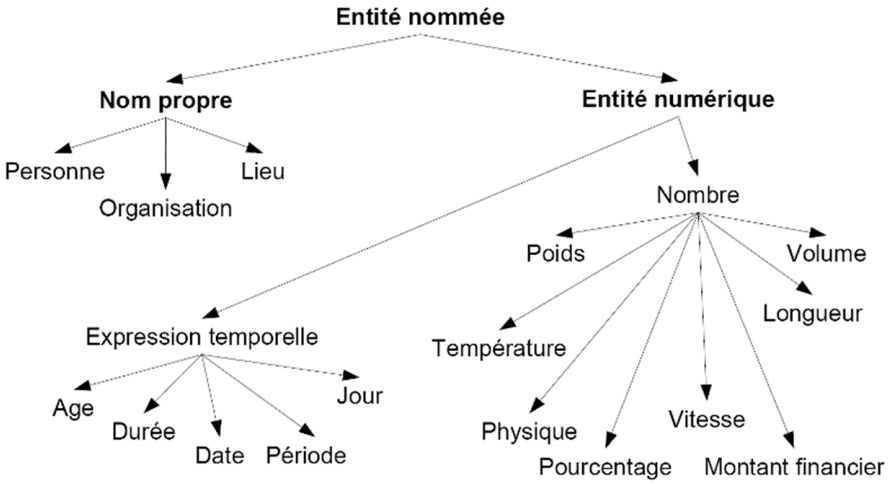
- colab.research.google.com/drive/1sJFiPuIEKQLQ-ZZX4tbJiHeNCGhU9_-V Notebook d'initiation à Spacy

Autres sites web

- La pseudonymisation par l'IA en pratique – ETALAB – gouv.fr
- ner-cli-custom-named-entity-recognition-with-spacy-in-four-lines blog codecentric 11/2020/
- Information Extraction With NLP And Deep Learning (nanonets.com)
- Extracting Data from Invoices with Google AutoML Natural Language (arthurkoziel.com)
- Amazon Sage Maker Named Entity Recognition - Amazon SageMaker
- Microsoft AZURE
    - Cognitive Services APIs Reference (microsoft.com)
    - Quickstart: Text mining using the Text Analytics client library - Azure Cognitive Services | Microsoft Docs

2 Synthèse

La reconnaissance d'entités nommées consiste à identifier des éléments qu'il est intéressant de pouvoir distinguer dans un corpus (ensemble de textes). Ces éléments peuvent être des noms propres de personnes, d'organisations ou de lieux, ainsi que des valeurs, des dates ou des heures.



Les algorithmes de NER permettent l'identification de ces entités, de les catégoriser et éventuellement de les normaliser comme dans l'exemple ci-dessous :

Identification	Lionel Jospin, jeudi 28 septembre, RTL, 2007.
Catégorisation	L'ancien premier ministre socialiste <PERS>Lionel Jospin</PERS> a confirmé, <DATE>jeudi 28 septembre</DATE>, sur <ORG>RTL</ORG>, qu'il ne sera pas candidat à l'investiture socialiste pour la présidentielle de <DATE>2007</DATE>.
Normalisation	L. Jospin -> Lionel Jospin

La reconnaissance d'entités nommées est une discipline du Traitement automatique du langage naturel (TALN). Le TALN ou Natural Language Processing (NLP) est un domaine multidisciplinaire impliquant la linguistique, l'informatique et l'intelligence artificielle, qui vise à créer des outils de traitement de la langue naturelle pour diverses applications.

Il existe de nombreuses applications du NER tel que l'indexation de documents, la traduction automatique, la pseudonymisation de documents contenant des données à caractère personnel (documents médicaux, décisions de justice) ou l'extraction automatiquement de données.

2.1 Le modèle

Le NER repose sur l'utilisation d'un modèle statistique qui appliqué à un texte par l'intermédiaire de fonctions fournies par une librairie de NLP permet d'identifier les entités souhaitées.

Ce modèle est généré par un processus d'entrainement qui à partir d'une grande quantité de données annotées apprend les formes possibles des entités nommées.

Il est possible d'importer des modèles existants qui ont été entrainés pour identifier les entités les plus courantes (personnes, lieux, dates …) et sur les langues les plus utilisées.

Les librairies ne proposent pas de « méta » modèle capable d'identifier toutes les entités dans toutes les langues et pour tous les sujets que pourrait aborder un corpus. Ceci pour de nombreuses raisons tels que la taille des modèles ou la perte de performances …

Par exemple le modèle « en_core_web_lg » importable dans Spacy permet d'identifier 18 entités différentes mais uniquement pour la langue anglaise. Il est entrainé uniquement sur des textes de blogs, de news, et des commentaires. Même optimisé sa taille est de 746 MB.

La création de modèles personnalisés s'avère nécessaire pour de multiples situations tels que :

- L' identification d'entités spécifiques tels que des numéros de SIRET d'entreprise
- Le traitement de documents avec un formalise particulier : bon de commande, email …
- Une langue ou un dialecte ne disposant d'aucun modèle
- L'obtention de scores (accuracy, F1-Score …) médiocres lors de l'évaluation de votre modèle

La plupart des techniques d'entrainement de modèles reposent sur le Deep Learning et peuvent être difficiles à mettre en œuvre. Ces algorithmes ont pour rôle d'identifier et d'extraire les règles du langage naturel en trouvant des patterns et des corrélations dans le corpus.

Les librairies de NLP implémentent de manière optimisé ces techniques et facilitent ainsi la création de ces modèles personnalisé (custom NER). Toutefois plusieurs milliers de documents annotés peuvent être nécessaire pour entrainer un modèle exploitable en production.

2.2 L'annotation

L'annotation consiste en une identification et labélisation manuelle des documents qui vont être utilisés pour l'entrainement.

La réalisation de cette tâche sur des milliers de documents et pour des dizaines de labels est très couteuse en temps. Il est souvent nécessaire de faire appel à des spécialistes disposant d'une bonne connaissance du sujet comme en médecine ou en droit.

Des outils d'annotations tels que Prodigy ou Doccano (liste sur GitHub mariananeves/annotation-tools) proposent une interface facilitant la labélisation et la classification en chaine de documents et génèrent des fichiers formatés pour les principaux Framework de NLP (eg JSON).

Pour réduire ce temps d'annotation ou pour compenser une insuffisance de documents, il est possible de générer automatiquement une partie des documents annotés via des procédés utilisant des ontologies, ou en intégrant des corpus pré-annotés ou par data augmentation. Ces techniques nécessitent un état de l'art et une veille à part entière

2.3 Les librairies

Les algorithmes de NER sont fournis pas des librairies proposant de multiples autres fonctions de traitement NLP, parmi celles-ci :

Flair et SpaCy présentent l'avantage de proposer des algorithmes à l'état de l'art tout en facilitant l'expérience utilisateur.

Flair est un framework simple pour le NLP. Il permet d'utiliser des modèles de NLP à l'état de l'art sur des textes de tout genre. Il est écrit en Python

SpaCy est un module Python à forte capacité d'industrialisation pour le NLP rédigée en Python et Cython. Il implémente les toutes dernières recherches dans le domaine du traitement du langage naturel et a été conçu pour être utilisé en production.

Stanza de l'Université de Stanford est une librairie Python qui utilise la librairie CoreNLP. Elle permet un entraînement et une évaluation efficace avec vos propres données annotées.

JohnSnowLabs, Apache OpenNLP, Google Auto ML sont également très populaires.

2.4 Performance

Les principaux indicateurs utilisés sont

- L'accuracy qui permet de connaître la proportion de bonnes prédictions par rapport à toutes les prédictions -> Nombre de bonnes prédictions / Nombre total de prédictions
- La précision correspond au nombre de documents correctement attribués à la classe i par rapport au nombre total de documents prédits comme appartenant à la classe i (total predicted positive).
- Le rappel (Recall) correspond au nombre de documents correctement attribués à la classe i par rapport au nombre total de documents appartenant à la classe i (total true positive)
- Le F1-Score combine la précision et le rappel. Le nombre de vrais négatifs (tn) n'est pas pris en compte.
- Les matrices de confusion (confusion matrix), sensibilité, spécificité, courbe ROC sont d'autres indicateurs pertinents

Beaucoup d'articles de benchmarking comparent les performances des librairies de NLP toutefois ces librairies évoluent rapidement et ces scores sont obtenus sur des corpus pouvant présenter des caractéristiques très éloignées de vos besoins.

Pour des modèles personnalisés il est nécessaire de surveiller les scores globaux du modèle et évaluer les scores pour chaque entité souhaité afin d'orienter les actions d'amélioration.

On peut améliorer en modifiant la labélisation, en ajoutant des prétraitements, en modifiant le nombre de documents d'entraiment …

2.5 Les défis

Les commentaires sur les forums et blog spécialisé soulèvent des difficultés rencontrées pour l'extraction d'entités issus de tableaux, ou sur la perte significative de performance sur les documents étant passés par un traitement d'OCRisation. Les entités ambiguës : métonymie et polysémie sont difficilement gérables. Par exemple dans la phrase « tout cela se décide à Bruxelles » parle-t-on de la ville ou de l'Union Européenne ?

3 Conclusion

La mise à disposition de librairies de NLP bénéficiant de l'état de l'art en matière de procédé d'entrainement combiné au déploiement d'outils d'annotations participent à l'essor de la création de modèles de NER personnalisés pour de nombreux cas d'usages.

Toutefois, peu de librairies présentent les caractéristiques nécessaire pour leur utilisation en production alors que les services cognitifs proposés en mode cloud par les GAFAM ne proposent pas de solution de création de modèles personnalisés.

La librairie Spacy présente des caractéristiques d'industrialisation intéressante : un rythme de développement soutenu et se classe parmi les plus performantes par de multiple benchmark. Une documentation officielle abondante et à jour. Le support très réactif tant en réponse aux tickets sur son repository GitHub que sur le forum officiel de l'éditeur. La communauté est très active avec de nombreuses publications de projet avec code source, des notebook (Google Collab) ou d'articles (medium) et tutoriaux vidéo (Youtube). Les dernières évolutions (v3) portent sur la customisation et

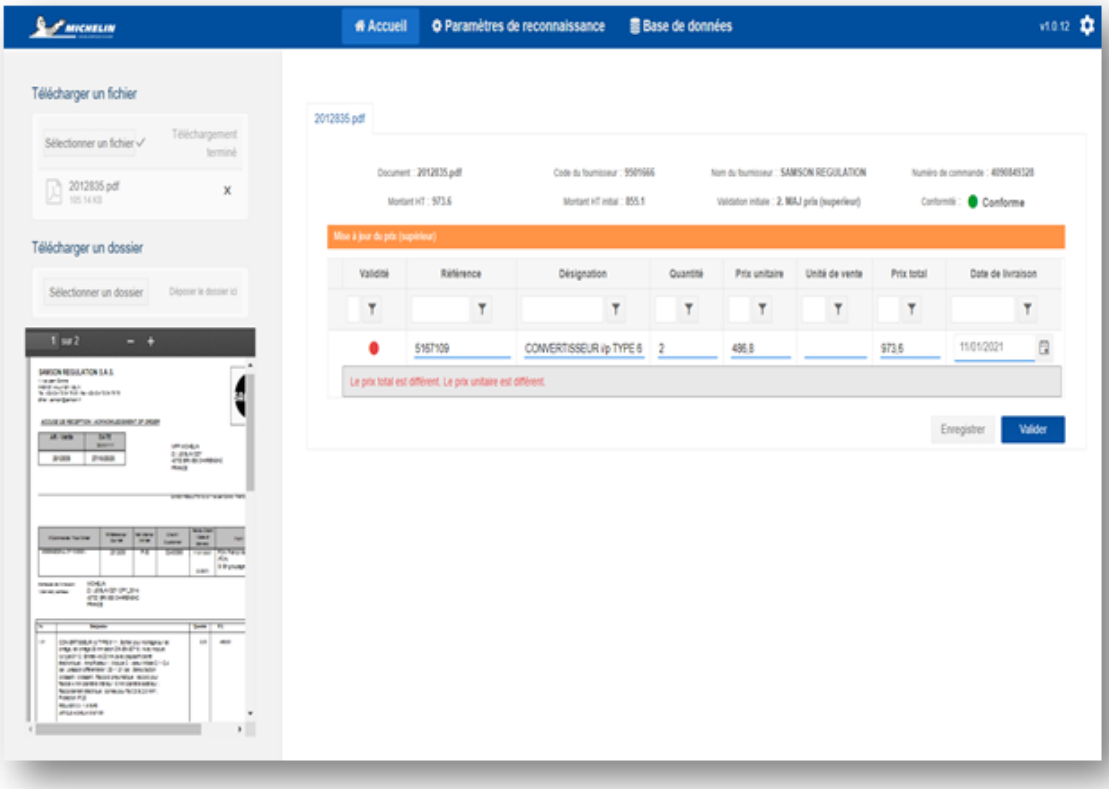
l’optimisation de l’accès aux ressources de calcul pour l’entrainement ce qui renforce son positionnement comme outil industriel.

L’activation d’une veille technologique quotidienne permet de rester vigilant aux évolutions des techniques de reconnaissance d’entités nommées, de leur intégration au sein des librairies de NLP tout en consultant de nouvelles mise en œuvre dans d’autres projets.

Le NLP reste un sujet de recherche très actif et certains points restent à améliorer tels que le traitement de documents océrisés ou des données en table.

Annexe 2. Form Analyzer

Capture d’écran du premier outil qui réalise l’extraction d’entités avec des regex et non de l’IA

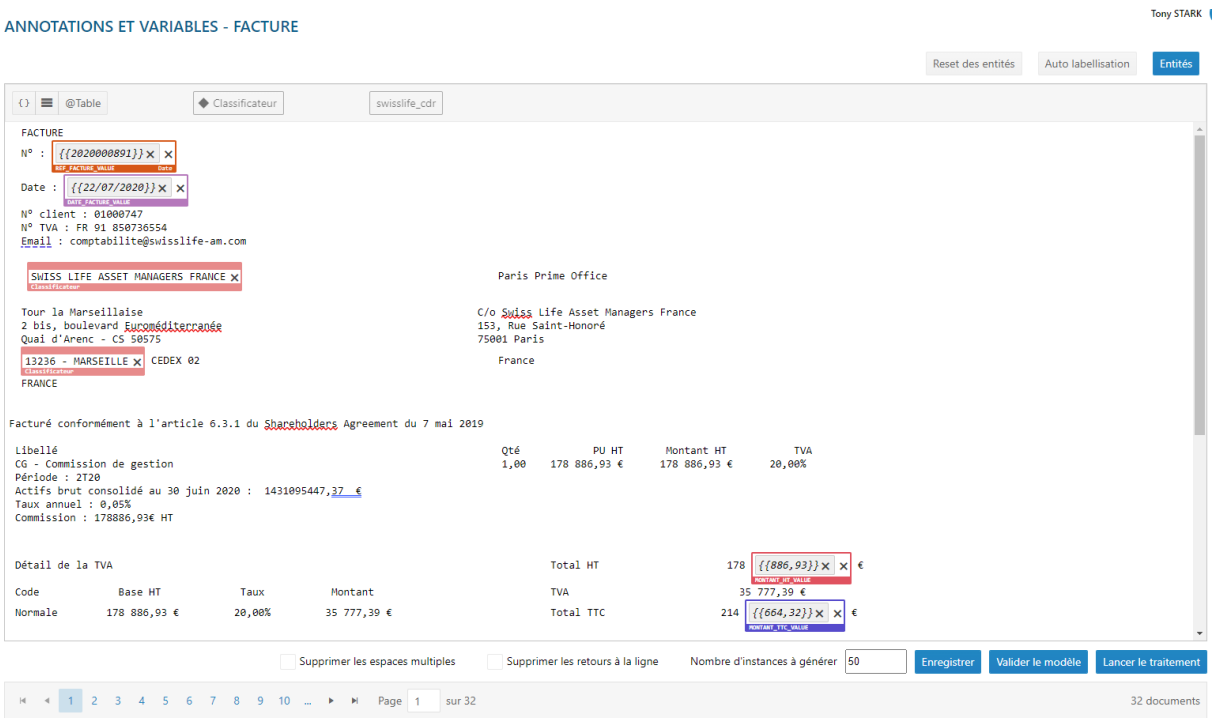


Annexe 3. Backlog Produit

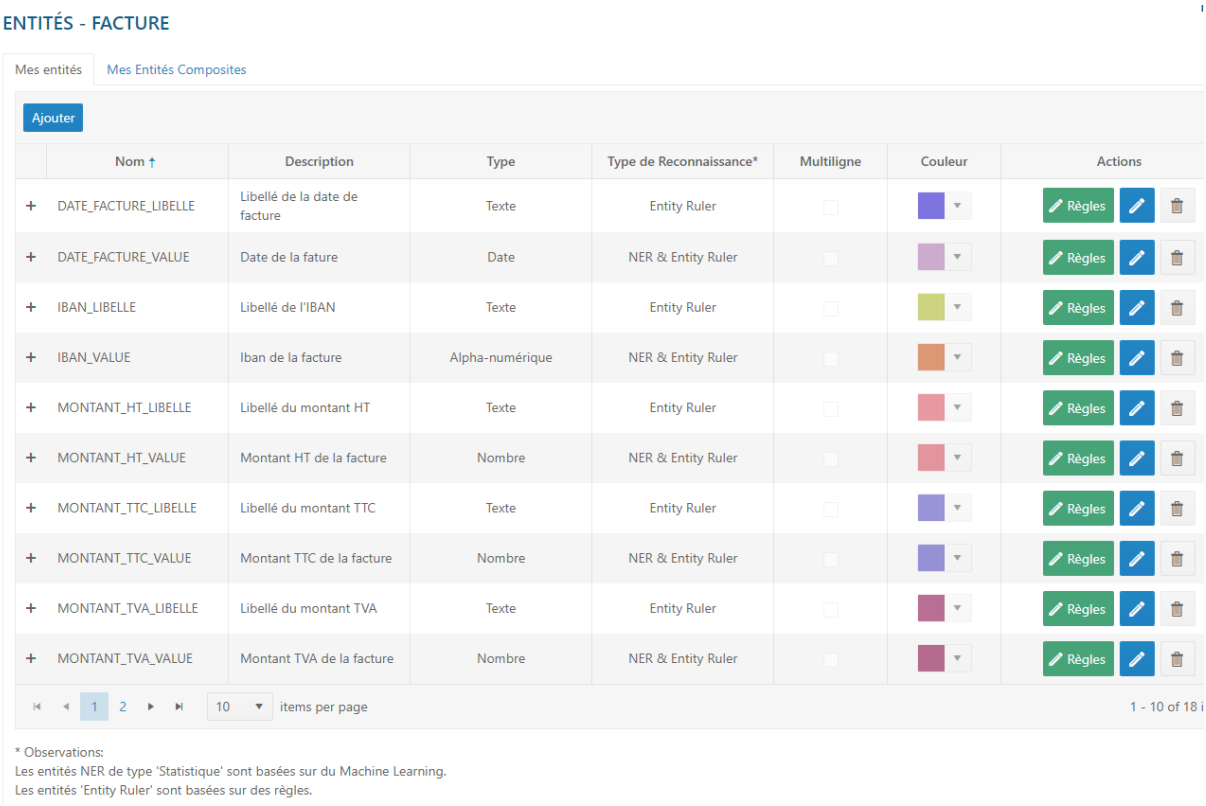
Le Product Backlog contient l'ensemble des fonctionnalités désirées par les utilisateurs, traduites principalement sous forme de petites histoires appelées User Stories.

User Story	Périmètre
En tant qu'utilisateur, je dois pouvoir m'authentifier afin d'utiliser l'application de manière sécurisée.	Général
En tant qu'utilisateur, je peux ajouter des membres.	Général
En tant que membre d'une équipe, je peux gérer les projets (Visualisation + Création).	Général
En tant que membre d'un projet, je peux gérer les templates (Visualisation + Création).	Général
En tant qu'utilisateur, je suis capable de créer des entités et de les utiliser.	Général
En tant qu'administrateur, je dois pouvoir visualiser l'état de fonctionnement de l'application afin de m'assurer de son bon fonctionnement	Admin
En tant qu'administrateur je dois pouvoir être alerté en cas de dysfonctionnement de l'application afin d'y remédier.	Admin
En tant qu'administrateur, je dois pouvoir utiliser des containeurs afin de déployer facilement l'application.	Admin
En tant qu'administrateur, je dois pouvoir déployer l'application afin de la rendre disponible aux utilisateurs.	Admin
En tant qu'utilisateur, je dois pouvoir entraîner facilement un modèle et visualiser ses performances	Entraînement
En tant qu'utilisateur, je dois pouvoir visualiser l'historique des entraînements.	Entraînement
En tant qu'utilisateur, je dois pouvoir enregistrer les informations des documents passés dans les modèles afin de pouvoir y avoir accès ultérieurement	Entraînement
En tant qu'utilisateur, je dois pouvoir consulter les indicateurs de performances de mes modèles.	Entraînement
En tant qu'utilisateur, je suis capable de générer un dataset à partir d'un document.	Entraînement
En tant qu'utilisateur, je dois pouvoir soumettre un document afin d'extraire les informations souhaitées.	Prédiction
En tant qu'utilisateur, je suis capable de récupérer la prédiction d'un modèle et l'exploiter dans l'interface graphique.	Prédiction
En tant qu'utilisateur, je dois pouvoir visualiser le texte et les entités prédites afin d'évaluer les résultats des prédictions et la pertinence du modèle.	Prédiction

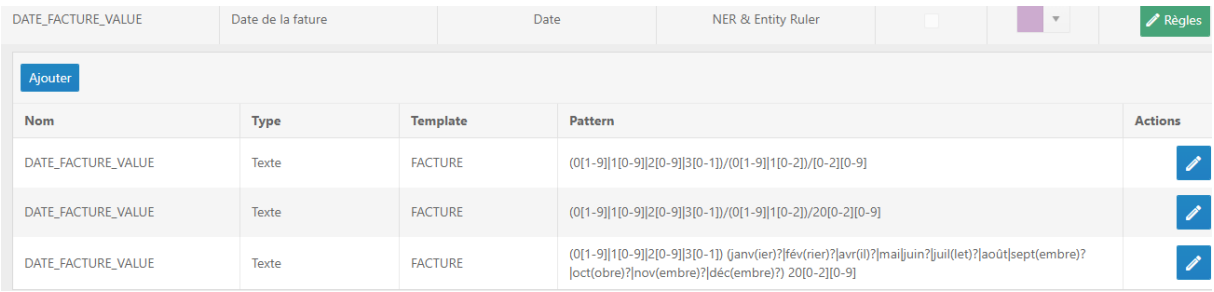
Annexe 4. Capture d'écran de l'interface d'annotation



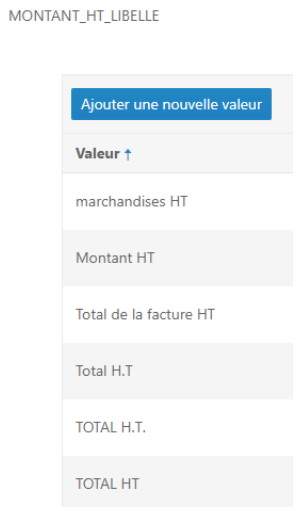
Annexe 5. Capture d'écran de l'interface de paramétrage des entités recherchées



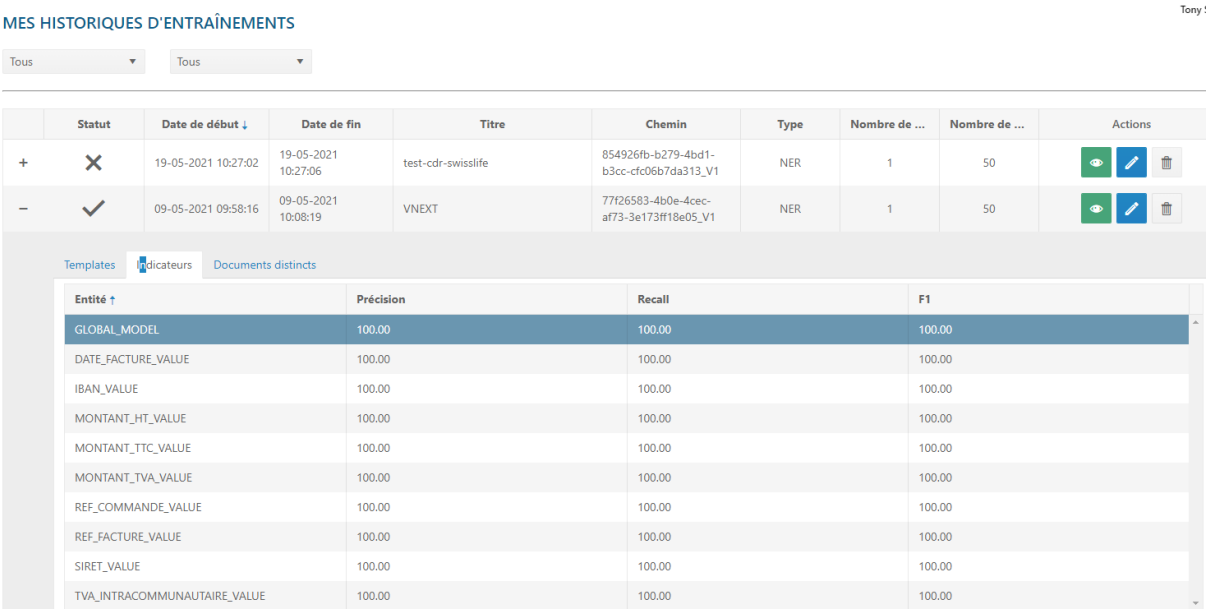
Annexe 6. Capture de l'interface de détails des paramètres d'une entité



Annexe 7. Capture de l'interface d'un dictionnaire pour une Entity Ruler

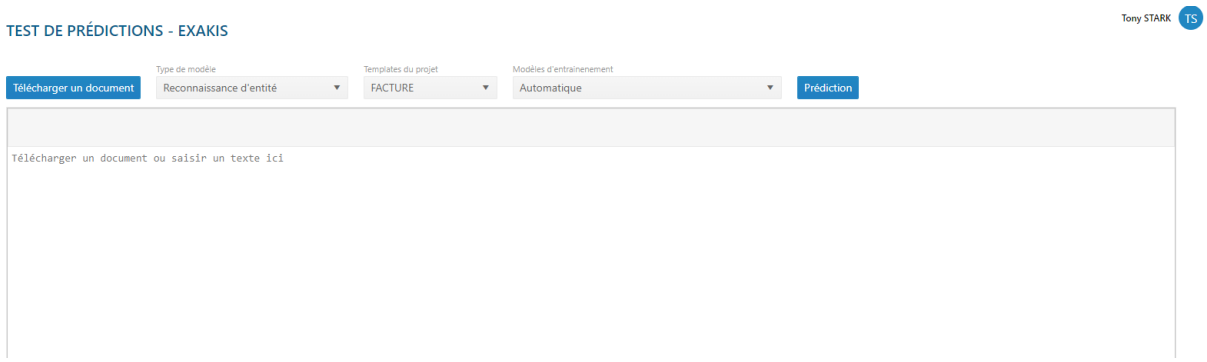


Annexe 8. Historique d'entrainement

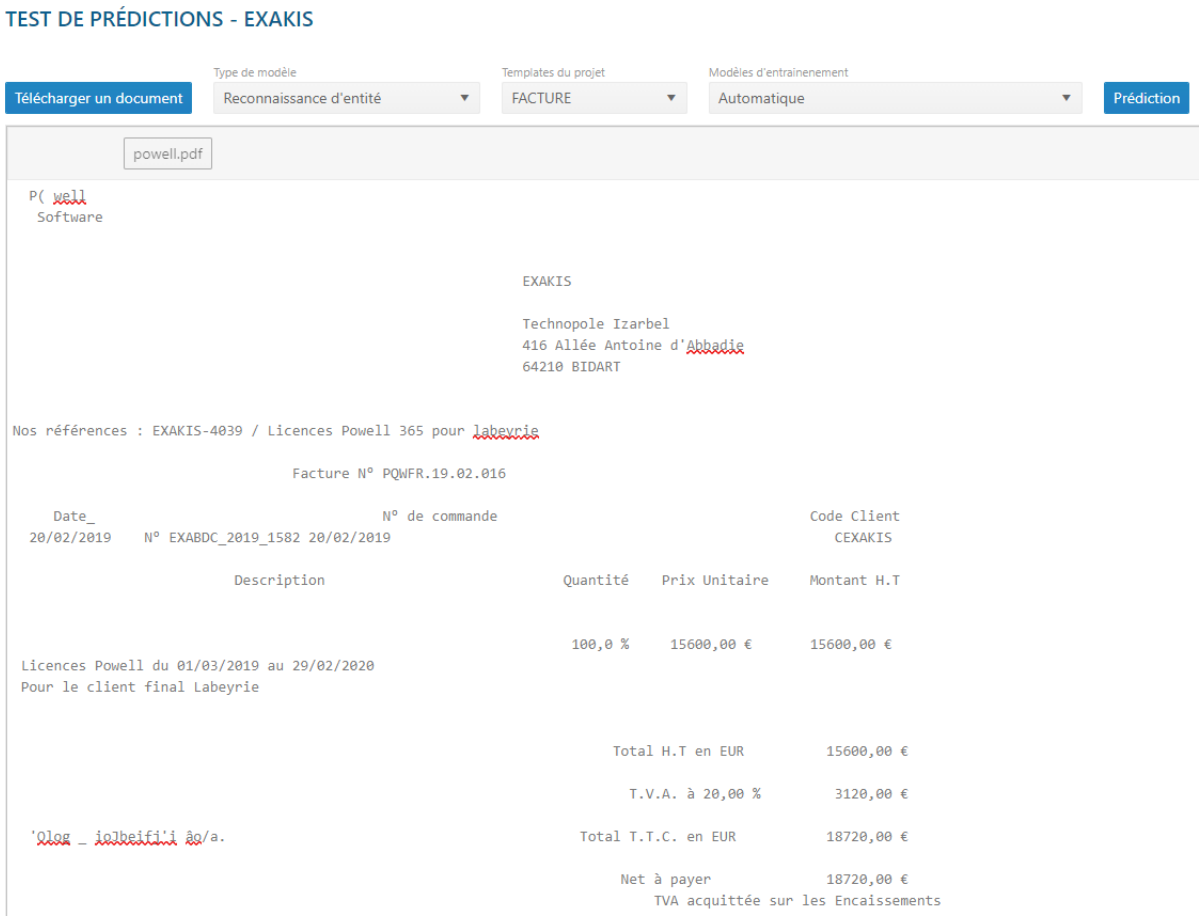


Annexe 9. Interface de prédiction

Interface avant l'import d'un document :



Affichage d'une facture importée au format PDF et océrisée :



Affichage des entités sur le document

Liste des entités trouvées avec le modèle générique

Entité	Valeur
REF_COMMANDE_LIBELLE	Nos références :
REF_COMMANDE_VALUE	EXAKIS-4039
REF_FACTURE_LIBELLE	Facture N°
REF_FACTURE_VALUE	PQWFR.19.02.016
DATE_FACTURE_LIBELLE	Date_
DATE_FACTURE_VALUE	20/02/2019
MONTANT_HT_LIBELLE	Total H.T
MONTANT_HT_VALUE	15600,00
MONTANT_TVA_LIBELLE	T.V.A. à 20,00
MONTANT_TVA_VALUE	3120,00
MONTANT_TTC_LIBELLE	Total T.T.C.
MONTANT_TTC_VALUE	18720,00
DATE_FACTURE_LIBELLE	le
DATE_FACTURE_VALUE	22/03/2019
IBAN_LIBELLE	IBAN
IBAN_VALUE	FR76 3000 4031 3000 01071115 665

Annexe 10. Insertion des données analytiques

Script C# côté backend .Net qui enregistre un fichier dans le FileShare via AppStorageService

```
// On génère les paths
string canvasHtmlPath = $"{labeledDocument.Document.Path}/Canevas/caneva.html";

// On enregistre l'html en tant que .html
AppStorageService.FileShare.SaveFile(user, canvasHtmlPath, labeledDocument.Html);

// On récupère les classifiers
labeledDocument.Document.Classifiers = string.Join("|", GetClassifiers(user, labeledDocument));

List<string> tagedEntities = GetEntities(user, labeledDocument);
labeledDocument.Document.EntityCount = tagedEntities.Count;
labeledDocument.Document.EntityList = string.Join("|", tagedEntities);
```

Annexe 11. Script d'entrainement

```
def train_spacy_ner(TRAIN_DATA,
                iterations):

   # We create a blank model
   nlp = spacy.blank('fr')

   # We create the ner pipe if not present
   if 'ner' not in nlp.pipe_names:
      ner = nlp.create_pipe('ner')
      nlp.add_pipe(ner, last= True)

   # We add all the labels to the model
   for _, annotations in TRAIN_DATA:
      for ent in annotations.get('entities'):
         ner.add_label(ent[2])

   # We disable other pipe to avoid to affect them
   other_pipes = [pipe for pipe in nlp.pipe_names if pipe != 'ner']

   # We start the train
   with nlp.disable_pipes('other_pipes'):
      optimizer = nlp.begin_training()
      for itn in range(iterations):
         s_time = process_time()
         random.shuffle(TRAIN_DATA)
         losses = {}
         for text, annotations in TRAIN_DATA:
            nlp.update([text],
                  [annotations],
                  drop= 0.5,
                  sgd = optimizer,
                  losses= losses
                  )
         exec_iter_time = process_time() - s_time
         progress = round(((itn / iterations) * 100), 2)
         print(f"Iter n° : {itn} -- Progress {progress} % -> Losses : {losses} -- Iteration execution time : {exec_iter_time}
seconds.")

   # #================================
   # # SAVE MODEL
   nlp.to_disk("./models/"+ "model_name")

   print(f'Modèle  enregistré')
```

Annexe 12. Script d'évaluation du modèle

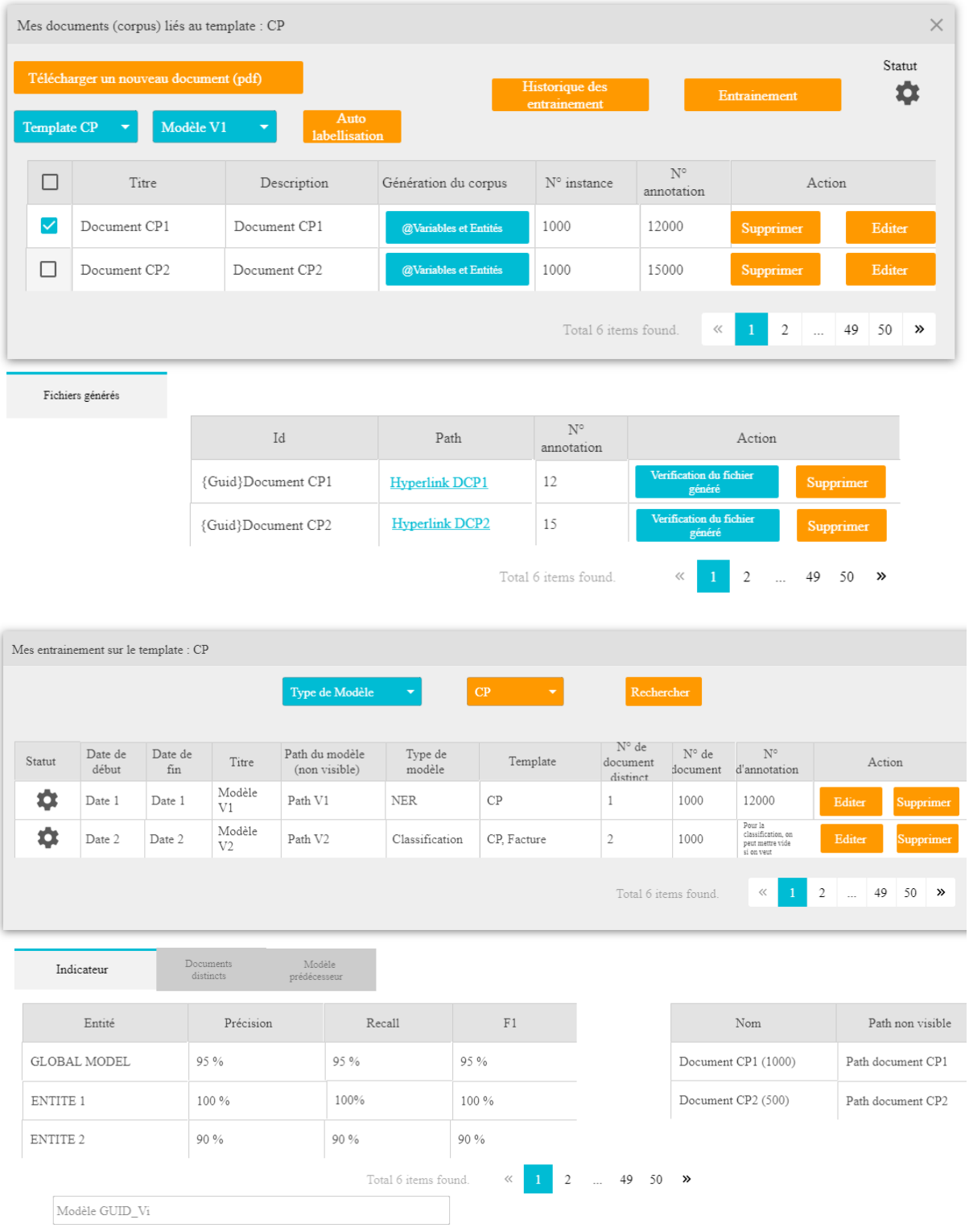
```
def evaluate_NER_Model(nlp, TEST_DATA):

    returnScores = []

    scorer = Scorer()
    for input_, annot in TEST_DATA:
        doc_gold_text = nlp.make_doc(input_)
        gold = GoldParse(doc_gold_text, entities=annot["entities"])
        pred_value = nlp(input_)
        scorer.score(pred_value, gold)
    returnScores.append(Scores(entity="GLOBAL_MODEL",
                        precision=np.around(np.mean(scorer.scores['ents_p']), 3),
                        recall=np.around(np.mean(scorer.scores['ents_r']), 3),
                        f1score=np.around(np.mean(scorer.scores['ents_f']), 3)))

    return returnScores[0]
```

Annexe 13. Routes Angular

Routes	Composant Angular	Fonction	Authent
client	LoginComponent	Authentification du client	Non
client/clients	ClientsComponent	Gestion/ Ajout de client	AuthGuard
client/members	MembersComponent	Gestion des membres d'un client	AuthGuard
client/projects	ProjectsComponent	Gestion des projets d'un client	AuthGuard
client/projects/templates	TemplatesComponent	Gestion des templates d'un projet	AuthGuard,ProjectGuard
client/projects/templates/documents	DocumentsComponent	Gestion des documents liés à un template	AuthGuard, ProjectGuard, TemplateGuard
client/projects/templates/entities	EntitiesComponent	Gestion des entités liés à un template	AuthGuard, ProjectGuard, TemplateGuard
client/dictionaries	DictionariesComponent	Gestion/ Ajout de dictionnaires	AuthGuard
client/dictionaries/:id	DictionaryComponent	Affichage d'un dictionnaire suivant son ID	AuthGuard
client/flows	FluxComponent	Gestion des flux.	AuthGuard
client/generated-files	GeneratedFilesComponent	Listing des fichiers générés	AuthGuard
client/importations	ImportationsComponent	Gestion des imports de documents	AuthGuard
client/labeler/:id	LabelerComponent	Outil de labélisation d'un document par son ID	AuthGuard, ProjectGuard, TemplateGuard
client/logging	LoggingComponent	Gestion des logs	AuthGuard
client/logging-details	LoggingDetailsComponent	Affichage des détails des logs	AuthGuard
client/predictions	PredictionComponent	Interface de prédiction	AuthGuard, ProjectGuard
client/trainings-history	TrainingsHistoryComponent	Historique des entrainements/ Affichage des modèles	AuthGuard, ProjectGuard
client/model/:modelId	TrainMonitorComponent	Monitoring d'un modèle	AuthGuard, ProjectGuard
client/training-details	TrainingDetailsComponent	Détails de l'entrainement d'un modèle	AuthGuard
client/error	ErrorComponent	Gestion des erreurs d'un projet	AuthGuard

Annexe 14. Mokup de l'IHM

Les maquettes de l'IHM sont réalisées par le PO (Product Owner) avec Pencil



Annexe 15. docker-compose.yaml

```
version: '3'

services:
  api:
    restart: always
    build:
      context: ./
      dockerfile: Dockerfile
    container_name: django
    volumes:
      - /fileshares/platform-storage:/platform-storage
    command: python /src/manage.py runserver 0.0.0.0:8000
    ports:
      - "8000:8000"
      - "2222:2222"
```

Annexe 16. Dockerfile du conteneur PredictionAPI

```
FROM python:3.8.2

ENV PYTHONUNBUFFERED 1
ENV DEBUG False
ENV SECRET_KEY 69e9d1c7-7b64-4b49-90c8-d808c8b10e34
ENV SHARE_NAME "/platform-storage/"

RUN mkdir /src
WORKDIR /src

ADD requirements.txt /src

RUN pip install -r requirements.txt

ADD . /src

RUN python ./manage.py collectstatic

CMD python manage.py runserver 0.0.0.0:8000

EXPOSE 8000
EXPOSE 2222
```

Annexe 17. Dockerfile du conteneur TrainAPI

```
FROM python:3.8.2

ENV PYTHONUNBUFFERED 1
ENV DEBUG False
ENV SECRET_KEY 53789566-896a-409b-a11e-48149fbbd7c7
ENV SHARE_NAME "/platform-storage/"

RUN mkdir /src
WORKDIR /src

ADD requirements.txt /src

RUN pip install -r requirements.txt

ADD . /src

RUN python ./manage.py collectstatic

CMD python manage.py runserver 0.0.0.0:8080

EXPOSE 8080
EXPOSE 2222
```

Annexe 18. Arborescence du stockage des fichiers sur Azure File Share

NOM_CLIENT

- LOG
  - FLUX 1
    - GUID_Document1.pdf
    - GUID_Document2.pdf
    - Etc…
- ACHAT
  - TEMPLATES
    - CP
      - Document CP1
        - SOURCE
          - Fichier PDF
          - Fichier PDF searchable
          - Fichier TXT
        - CANEVAS
          - Fichier variabilisé et annoté
        - INSTANCES
          - Fichiers au format JSONL
        - SAMPLES
          - Fichiers au format JSONL
      - Document CP2
        - SOURCE
          - Fichier PDF
          - Fichier PDF searchable
          - Fichier TXT
        - CANEVAS
          - Fichier variabilisé et annoté
        - INSTANCES
          - Fichiers au format JSONL
        - SAMPLES
          - Fichiers au format JSONL
  - MODELES IA
    - NER
      - CP
        - ENTITY_RULER
          - DATE_FATCURE.json
          - N_CLIENT.json
        - VERSIONS
          - Guid_V1
          - Guid_V2
    - CLASSICATION
      - VERSIONS
        - Guid_V1
        - Guid_V2
    - LOG

Annexe 19. Script SQL de création de la base de données

Script executé dans l'outil Microsoft SSMS (SQL Server management Studio) qui permet de créer des base de type Microsoft SQL Server

```
USE master ;
GO
CREATE DATABASE DematIA
ON
( NAME = DematIA_db,
   FILENAME = 'C:\Program Files\Microsoft SQL
Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\dematia_db.mdf',
   SIZE = 10,
   MAXSIZE = 50,
   FILEGROWTH = 5 )
LOG ON
( NAME = DematIA_log,
   FILENAME = 'C:\Program Files\Microsoft SQL
Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\dematia_log.ldf',
   SIZE = 5MB,
   MAXSIZE = 25MB,
   FILEGROWTH = 5MB ) ;
GO
```

Annexe 20. Script SQL de création des tables

```
-- Demat IA tables
-- Table: clients
CREATE TABLE DematIA.dbo.clients (
   id int identity(1,1) primary key,
   creation_date datetime  NOT NULL,
   connection_string nvarchar(250)  NOT NULL,
   description nvarchar(500)  NOT NULL,
   email nvarchar(250)  NOT NULL,
   host_prediction_api nvarchar(250)  NOT NULL,
   host_train_api nvarchar(250)  NOT NULL,
   title nvarchar(250)  NOT NULL,
);

-- Table: documents
CREATE TABLE DematIA.dbo.documents (
   id int identity(1,1) primary key,
   creation_date datetime  NOT NULL,
   entity_list nvarchar(250)  NOT NULL,
   classifier nvarchar(250)  NOT NULL,
   title nvarchar(250)  NOT NULL,
   path nvarchar(250)  NOT NULL,
   instance_count int  NOT NULL,
   entity_count int  NOT NULL,
);

-- Table: projets
CREATE TABLE DematIA.dbo.projets (
   id int  identity(1,1) primary key,
   creation_date datetime  NOT NULL,
   title nvarchar(250)  NOT NULL,
```

```
   client_id int  NOT NULL,
        CONSTRAINT Projets_Clients FOREIGN KEY (client_id) REFERENCES DematIA.dbo.clients (id)
);

-- Table: templates
CREATE TABLE DematIA.dbo.templates (
   id int identity(1,1) primary key,
   creation_date datetime  NOT NULL,
   description nvarchar(250)  NOT NULL,
   project_id int  NOT NULL,
   title nvarchar(250)  NOT NULL,
   type nvarchar(250)  NOT NULL,
        CONSTRAINT Projets_Templates FOREIGN KEY (project_id) REFERENCES DematIA.dbo.projets (id)
);

-- Table: models
CREATE TABLE DematIA.dbo.models (
   id int identity(1,1) primary key,
   creation_date datetime  NOT NULL,
   end_date datetime  NOT NULL,
   is_generic bit  NOT NULL,
   number_distinct_documents int  NOT NULL,
   path nvarchar(250)  NOT NULL,
   predecessor_model nvarchar(250)  NOT NULL,
   status nvarchar(250)  NOT NULL,
   template_id int  NOT NULL,
        CONSTRAINT models_templates FOREIGN KEY (template_id) REFERENCES DematIA.dbo.templates (id)
);

-- Table: model_document
CREATE TABLE DematIA.dbo.model_document (
   id int identity(1,1) primary key,
   creation_date datetime  NOT NULL,
   model_id int  NOT NULL,
   documents_id int  NOT NULL,
   models_id int  NOT NULL,
        CONSTRAINT model_document_documents FOREIGN KEY (documents_id) REFERENCES
DematIA.dbo.documents (id),
        CONSTRAINT model_document_models FOREIGN KEY (models_id) REFERENCES DematIA.dbo.models
(id)
);
```

Annexe 21. Script SQL d'insertion de données

Dans la table clients

```
INSERT INTO [DematIA].[dbo].[clients]
        ([creation_date]
        ,[connection_string]
        ,[description]
        ,[email]
        ,[host_prediction_api]
        ,[host_train_api]
        ,[title])
    VALUES
        (GETDATE()
        ,'assursafeEndPointSpecimen'
        ,'Client spécialisé dans les assurances'
        ,'m.dupont@assursafe.fr'
        ,'assursafe.predict_url'
        ,'assursafe.train_url'
        ,'Assursafe'),
                        (GETDATE()
        ,'AxaEndPointSpecimen'
        ,'Client spécialisé dans les assurances'
        ,'m.dupont@axa.fr'
        ,'axa.predict_url'
        ,'axa.train_url'
        ,'Axa'),
                        (GETDATE()
        ,'MichelinEndPointSpecimen'
        ,'Fournisseur automobile'
        ,'m.dupont@michelin.fr'
        ,'michelin.predict_url'
        ,'michelin.train_url'
        ,'Michelin')
```

Dans la table projets

```
INSERT INTO [DematIA].[dbo].[projets]
        ([creation_date]
        ,[title]
        ,[client_id])
    VALUES
        (GETDATE(),
                        'Facture',
                        (SELECT id FROM DematIA.dbo.clients WHERE title='Axa')
                        )
```

Annexe 22. Script de tests

```
namespace AppStorageTests
{
   [TestClass]
   public class AppStorageServiceTests
   {
      // Utilisateur de test
      private User _User = new User()
      {
         Email = "test@test.com",
         Firstname = "test",
         Lastname = "test",
         Name = "test Test",
         Role = Role.GLOBAL_ADMIN,
         Id = "fake_id",
         NavigationContext = new NavigationContext()
         {
            ConnectionString = TestConstants.ClientConnectionString
         }
      };

      private TestData _TestData = new TestData()
      {
         RowKey = "8af98bb2-73ba-45ca-a651-05720d3d9832",
         Title = "Test"
      };

      [TestMethod]
      public void TableTest()
      {
         // Création
         Assert.IsNotNull(AppStorageService.TableStorage.Save(_User, _TestData));

         // Récupération
         Assert.AreEqual(AppStorageService.TableStorage.Get<TestData>(_User, _TestData.RowKey)?.Title, _TestData.Title);

         // Récupération avec filtre
         Assert.AreEqual(AppStorageService.TableStorage.GetByFilter<TestData>(_User, $"RowKey eq '{_TestData.RowKey}'")?.Title,
_TestData.Title);

         // Suppression de la données
         Assert.IsTrue(AppStorageService.TableStorage.Delete<TestData>(_User, $"RowKey eq '{_TestData.RowKey}'"));

         // Suppression de la table
         Assert.IsTrue(AppStorageService.TableStorage.DropTable<TestData>(_User));
      }

      [TestMethod]
      public void FileTest()
      {
         AppStorageService.FileShare.FileShareName = TestConstants.FileShareName;

         // Création
         Assert.IsTrue(AppStorageService.FileShare.SaveFile(_User, "test-folder/sub/test.txt", "Ceci est un test unitaire"));

         // Récupération
         Assert.AreEqual(AppStorageService.FileShare.GetFile(_User, "test-folder/sub/test.txt"), "Ceci est un test unitaire");

         // Suppression du fichier
         Assert.IsTrue(AppStorageService.FileShare.DeleteFile(_User, "test-folder/sub/test.txt"));

         // Suppression du dossier
         Assert.IsTrue(AppStorageService.FileShare.DeleteFolderFiles(_User, "test-folder"));

         AppStorageService.FileShare.FileShareName = null;
      }
```

```
      [TestMethod]
      public void QueueTest()
      {
        // Création
        Assert.IsTrue(AppStorageService.QueueStorage.CreateQueue(_User, TestConstants.QueueName));

        // Insertion
        Assert.IsTrue(AppStorageService.QueueStorage.PushMessage(_User, "Ceci est un test que queue",
TestConstants.QueueName));

        // Récupération
        Assert.AreEqual(AppStorageService.QueueStorage.PullMessage<string>(_User, TestConstants.QueueName), "Ceci est un test
que queue");

        // Suppression du fichier
        Assert.IsTrue(AppStorageService.QueueStorage.DropQueue(_User, TestConstants.QueueName));
      }

      [Table("tests")]
      [DefaultPK("test")]
      public class TestData : TableEntity
      {
        [Key]
        public string Title { get; set; }
      }
    }
}
```

Annexe 23. Les outils

- VSCode
- Visual Studio pro
- CLI Azure
- Microsoft Azure Storage Explorer
- Docker
- Windows WSL 2
- Git
- Postman
- Azure DevOps

Le PO réalise des maquettes d'interfaces avec l'outil Pencil

Framework Spacy, Angular telerik .Net, Aspose, ABBY, Node