



MONTPELLIER INTERDISCIPLINARY CENTER  
ON SUSTAINABLE AGRI-FOOD SYSTEMS  
SOCIAL AND NUTRITIONAL SCIENCES



# **Expérimentation de l'interface DSW au sein d'une unité de recherche :**

Quelle transposition possible au niveau d'un institut et  
sous quelles conditions ?

Rapport de Stage pour la 2<sup>e</sup> année de mon BUT informatique en  
Réalisation d'applications : conception, développement et  
validation

Réalisé par  
**PICOT** Théodore

Sous la supervision de  
**Rousselle** Jean-Marc et **Aubert** Magali

Année universitaire 2022-2023

# Remerciements

Je tiens tout d'abord à remercier les unités MoISA et CEE-M du département EcoSocio de INRAE de m'avoir chaleureusement accueilli au sein de leur équipe et leurs locaux.

Je tiens également à remercier mes tuteurs de stage, Jean-Marc Rousselle et Magali Aubert, de m'avoir accompagné, conseillé et aidé tout au long de mon stage.

Je remercie également Soraya Lutete, Jonathan Mineau et Marek Suchanek, de m'avoir aidé et conseillé lors de la réalisation des missions du stage.

Je souhaite aussi remercier Alain Marie-Jeanne pour avoir tutoré ce stage.

Finalement, je tiens à remercier mon collègue et camarade de stage Jocelin Sanchez avec qui j'ai fortement collaboré pour la réalisation du stage.

# Résumés

## Résumé du rapport en français

L'objectif de ce stage est de développer des extensions pour un outil de rédaction de PGD : DSW. Un outil en phase de test au sein de l'unité MoISA qui fait partie d'INRAE.

Un PGD est un document évolutif qui détaille la manière dont les données sont gérées. Ce document est un élément clé permettant de prouver sa conformité aux exigences réglementaires du RGPD.

DSW propose des fonctionnalités et des outils puissants aux développeurs. Nous avons essayé d'exploiter ces fonctionnalités au maximum de leur potentiel, ce qui nous a facilité la réalisation de nos missions.

Mots-clés : RGPD, PGD, DSW, FAIR, Conformité.

## Abstract in english

The main goal of this internship is to undertake the development of extensions for DSW, an emerging tool for creating DMPs (Data Management Plans).

DMPs provide a detailed insight into data management practices and can serve as crucial evidence for demonstrating compliance with regulatory requirements, such as the General Data Protection Regulation (GDPR).

DSW strives to be flexible and iteroperable by offering a wide range of powerful features and tools to developers. By leveraging the extensive functionalities provided by DSW, we have found it considerably easier to accomplish our assigned tasks and fulfill the objectives of our internship.

# Sommaire

<b>Résumés</b>	<b>2</b>
Résumé du rapport en français	2
Abstract in english	2
<b>1. Introduction</b>	<b>6</b>
<b>2. Présentation d'INRAE et de MoISA</b>	<b>9</b>
<b>3. Présentation de Data Stewardship Wizard</b>	<b>11</b>
3.1. Le Knowledge Model (KM)	11
3.2. Le Questionnaire	12
3.2.1. Options Question	13
3.2.2. List Question	13
3.2.3. Value Question	15
3.2.4. Multi-choice Question	16
3.3. Le Document Template	16
3.4. Le Document	16
3.5. DSW vs DMP OPIDoR	16
3.5.1. DSW	16
3.5.2. DMP OPIDoR	18
<b>4. Analyse</b>	<b>19</b>
4.1. Exportation du PGD	19
4.2. Stockage du PGD sous Nextcloud	21
4.3. Nextcloud	22
4.4. Portée du stage	23
<b>5. Rapport technique</b>	<b>25</b>
5.1. Apprentissage et consolidation de DSW	25
5.1.1. Structure de DSW sous Docker	25
5.1.2. Test de l'API DSW	30
5.2. Développement du format CSV	31
5.2.1. Le format CSV	31
5.2.2. DSW-TDK	31
5.2.3. Version générique du format CSV	32
5.2.4. Version définitive du format CSV	38
5.3. Service de stockage sous Nextcloud	41
5.3.1. Test de l'API Nextcloud	41
5.3.2. Développement du service	42
<b>6. Méthodes de travail et gestion du projet</b>	<b>48</b>
6.1. Trello et Scrum	48
6.2. Création de notre organisation GitHub	49
<b>Conclusion</b>	<b>51</b>
<b>Bibliographie</b>	<b>53</b>
<b>Annexes</b>	<b>55</b>
docker-compose.yml de dsw-deployment-example	55

# Glossaire

**ANR** Agence nationale de la recherche

**API** Application Programming Interface. Une API est un ensemble de règles et de protocoles qui permettent à différentes applications de communiquer et d'échanger des données entre elles de manière structurée.

**CEE-M** Center for Environmental Economics - Montpellier

**CIRAD** Centre de coopération internationale en recherche agronomique pour le développement

**CIRM** Centre international de rencontres mathématiques

**CNRS** Centre national de la recherche scientifique

**CSV** Comma-Separated Values, CSV est un format de fichier simple utilisé pour stocker et échanger des données tabulaires.

**DMP OPIDoR** Data Managment Plan pour une Optimisation du Partage et de l'Interopérabilité des Données de la Recherche. Outil franco-français facilitant la création de PGD. Cet outil est largement utilisé au sein d'INRAE et d'autres instituts français.

**DOCX** Document Microsoft Word

**DPO** Data Protection Officer / Déléguée à la Protection des Données. Le rôle d'un DPO est de garantir la conformité et la protection des données personnelles au sein d'une organisation en conseillant, surveillant et sensibilisant. Son objectif principal est de préserver la confidentialité, l'intégrité et la disponibilité des données personnelles, en veillant au respect des lois et réglementations sur la protection des données.

**DSW** Data Stewardship Wizard, outil open source et international facilitant la création de PGD. Il est actuellement en phase de test au sein des unités MoISA et CEE-M.

**FAIR** Findable Accessible Interoperable Reusable. Quatre principes de la science ouverte cherchant à faciliter le traitement des données effectué par les machines.

**HCERES** Haut Conseil de l'Évaluation de la Recherche et de l'Enseignement Supérieur. Un haut conseil chargé d'évaluer les établissements de recherches.

**IDE** Integrated Development Environment. Terme désignant un éditeur de texte puissant et possédant de nombreuses fonctionnalités facilitant la rédaction de code et le développement d'application.

**INIST** Institut de l'information scientifique et technique

**INRA** Institut national de la recherche agronomique

**INRAE** Institut National de Recherche pour l'Agriculture, l'Alimentation et l'Environnement

**IRSTEA** Institut national de recherche en sciences et technologies pour l'environnement et l'agriculture

**MoISA** Montpellier Interdisciplinary center on Sustainable Agri-Food Systems

**PDG** Plan de Gestion de Données. Document évolutif contenant un plan définissant comment les données utilisées et générées lors d'un projet de recherches vont être gérées.

**RDO** Référent Données Opérationnel. Est responsable de la gestion des données essentielles pour les opérations de l'organisation, en veillant à leur qualité et leur sécurité.

**RGPD** Règlement Général sur la Protection des Données. Une loi européenne pour protéger les données à caractère personnelles. Il impose des règles strictes aux organisations sur la collecte et le traitement de ces données.

**SGBD** Système de Gestion de Base de Données

**Stata** Logiciel de statistiques et d'économétrie largement utilisé par les économistes et les épidémiologistes.

**UMR** Unité mixte de recherche

**UUID** Universally Unique IDentifier. Un identifiant unique universellement reconnu. C'est une chaîne de caractères alphanumériques qui permet d'identifier de manière unique une entité ou une ressource sans risque de collision avec d'autres identifiants



# 1. Introduction

En 2018, l'entrée en vigueur du Règlement Général sur la Protection des Données (RGPD\*) a imposé de nouvelles contraintes liées au traitement de données à caractère personnel. Des données sont dites à caractère personnel si elles permettent une identification directe ou indirecte des personnes enquêtées. Le traitement, au sens RGPD du terme, fait référence à l'ensemble des étapes du cycle de vie de la donnée, depuis sa collecte jusqu'à sa destruction ou son archivage. Les peines encourues en cas de non-conformité au règlement sont nombreuses, allant de sanctions pénales à de lourdes amendes financières pouvant s'élever à 4 % du chiffre d'affaires mondial de l'entreprise concernée. Au-delà de ces peines financières, la réputation des entreprises est une sanction immatérielle aux conséquences graves. Pour prouver la bonne conformité de leurs pratiques de traitement de leurs données avec le RGPD, les entreprises renseignent un Plan de Gestion de Données (PGD\*).

Le PGD est un document construit par les instituts de recherche, les agences de financement ou des juristes. Il détaille la manière dont les données sont collectées, stockées, utilisées, partagées, sauvegardées, anonymisées, détruites ou archivées. Ce document est ainsi un outil organisationnel évolutif qui se construit en début de projet et s'actualise tout au long de sa réalisation. C'est ainsi un outil clé pour garantir que les données sont traitées en conformité avec les exigences réglementaires du RGPD. Cela permet par ailleurs de renforcer la confiance des clients et des partenaires envers l'entreprise, en démontrant son engagement à protéger l'anonymat des personnes enquêtées ainsi que la confidentialité et la sécurité des données à caractère personnel qui sont collectées.

Au sein des instituts de recherche, le respect des bonnes pratiques en matière de gestion de données est crucial. Il en va de la réputation de ces instituts et donc son attractivité, ainsi de leur engagement dans la dynamique établie dans le cadre de la Science Ouverte. La traçabilité, la reproductibilité ou encore la fiabilité des données des recherches sont au cœur de cette ouverture des données de la recherche pour permettre à l'ensemble de la communauté scientifique de bénéficier des travaux précédemment menés. Dans certains cas, les données collectées peuvent être qualifiées de sensibles (données de santé, données relatives aux orientations sexuelles...) et leur diffusion non contrôlée peut engendrer des risques importants pour les personnes enquêtées. La justification de bonnes pratiques en matière de gestion de données est essentielle pour garantir la crédibilité des instituts de recherche.



Pour pallier ce risque, le [décret n° 2021-1572 du 3 décembre 2021](#) a été mis en place. Il impose de nouvelles pratiques par rapport à la conservation, la communication et la réutilisation des résultats des travaux scientifiques menés par les organismes de recherche. De plus, pour assurer la qualité des travaux menés par les entités de recherche, un comité d'expertise appelé le HCERES\*, en charge notamment de qualité des travaux de recherche, porte une attention particulière à cette mise en conformité.

C'est ainsi que le PGD est devenu un élément clé et une pratique essentielle pour justifier la qualité et la rigueur des recherches menées au sein des instituts de recherche. Le PGD doit être établi dès le début d'un projet et doit inclure le cycle de vie des données. Dans le cadre de données qui ne sont pas à caractère personnel, ce PGD va au-delà de la durée du projet en prévoyant les conditions de leur réutilisation. Ainsi, le PGD est considéré comme un élément crucial pour produire des données conformes aux principes FAIR\*. Dans le cadre de données dites à caractère personnel, le PGD est construit autour du principe n°3 de la Science Ouverte : les données doivent être *“aussi ouvertes que possible, aussi fermées que nécessaire”*.

Pour prouver la conformité de leurs pratiques avec les exigences réglementaires du RGPD, les chercheurs doivent fournir différents documents dont le PGD. INRAE\*, institut de recherche dans lequel j'ai réalisé mon stage, a trouvé des solutions informatiques qui facilitent la rédaction du PGD. Actuellement, l'outil le plus utilisé au sein d'INRAE est **DMP OPIDoR\***, service mis à disposition par l'INIST\* CNRS\* pour les établissements de recherches et d'enseignement.

L'UMR\* MoISA\* où j'ai travaillé a privilégié l'exploration d'un autre outil appelé DSW\*. Cet outil est actuellement en phase de test au sein de cette unité pilote. Je développerai davantage cet outil (Cf. [Page 11](#)).

L'objectif principal de ce projet est d'améliorer et d'ajouter certaines fonctionnalités à la version existante DSW. L'enjeu est pluriel : pour les chercheurs, il s'agit de faciliter la rédaction du PGD ; pour l'unité, il s'agit d'exploiter l'ensemble des PGD saisis ; pour la DPO\*, il s'agit d'interagir plus facilement avec les chercheurs pour faciliter le processus de mise en conformité ; pour les financeurs, il s'agit de disposer d'un document standardisé qui aborde l'ensemble des items nécessaires à leur exploitation.

Dans la suite de ce rapport, je vais aborder plusieurs points clé essentiels pour comprendre mon travail. Tout d'abord, je vais vous fournir une explication détaillée de DSW en expliquant la structure de cet outil et ses fonctionnalités. Ensuite, je me pencherai sur l'outil Nextcloud, un outil de stockage cloud, en précisant l'importance de certaines de ses fonctionnalités. Ceci sera développé dans la partie analyse du présent rapport.

Par ailleurs, une partie importante de ce rapport est l'analyse des missions et de l'existant. Il est important de noter que deux stages ont été conduits sur cette thématique. Je vous présenterai les différentes missions qui nous ont été confiées lors de ce stage en soulignant quelles ont été plus précisément les miennes et quelles interactions ont été développées avec Jocelin Sanchez, le second stagiaire. Après cette partie analyse, je décrirai les résultats obtenus dans le cadre des missions confiées. Je mettrai en évidence les réalisations liées à l'automatisation de l'exportation des PGD dans des formats variés, ainsi que l'intégration de Nextcloud pour le stockage et la gestion centralisée des PGD.

Enfin, je vous exposerai la méthodologie et l'organisation de ce projet. Je détaillerai les étapes suivies, les outils utilisés et les approches adoptées pour mener à bien les différentes missions.

## 2. Présentation d'INRAE et de MoISA

Au cours de mon stage, j'ai effectué mon travail au sein d'INRAE, un institut français de recherche spécialisé dans les domaines de l'agriculture, de l'alimentation et de l'environnement. Plus spécifiquement, j'ai travaillé au sein de l'unité MoISA, qui est l'une des nombreuses unités de recherche du département EcoSocio de INRAE. Je vais vous présenter brièvement ces deux entités.

INRAE est un établissement public français de recherche en sciences agronomiques, alimentaires et environnementales. Il a été créé en 2020 suite à la fusion d'INRA\* et d'IRSTEA\*. L'institut mène des recherches dans de nombreux domaines tels que l'agriculture, l'élevage, l'alimentation, la santé animale et végétale, l'environnement, la biodiversité, l'économie, les sciences sociales, les mathématiques et le numérique. L'objectif principal d'INRAE est de développer des connaissances scientifiques et des innovations technologiques pour répondre aux défis actuels liés à l'agriculture durable, la sécurité alimentaire, la santé environnementale et le changement climatique.

INRAE dispose de plusieurs sites de recherche en France et collabore avec d'autres institutions de recherche et d'enseignement supérieur dans le monde entier. Il est organisé en différents départements, chacun se concentrant sur un domaine de la recherche. Il y a 13 départements qui regroupent des unités de recherche, dont environ 200 réparties sur l'ensemble du territoire français.

L'UMR MoISA est une unité de recherche scientifique interdisciplinaire qui s'intéresse à l'analyse des transitions des systèmes agroalimentaires à travers le monde. Les recherches menées par l'UMR MoISA portent sur les interactions entre les systèmes d'approvisionnement, les environnements alimentaires, les comportements alimentaires, leurs résultats et leurs facteurs de changement. L'UMR MoISA se concentre sur les « drivers » politiques et socio-économiques, alors que d'autres unités de recherche travaillent sur les « drivers » environnementaux et démographiques.

Le projet scientifique de l'UMR MoISA vise à proposer des solutions pour aller vers des transitions souhaitables, notamment dans les pays du bassin méditerranéen, l'Afrique sub-saharienne, l'Asie du Sud-est et les Outre-mer.

L'unité de recherche est spécialisée dans les enquêtes, qualitatives et quantitatives, avec des formations organisées sur la qualité des données. L'animation scientifique de l'UMR MoISA s'articule autour de 3 pôles de recherche :

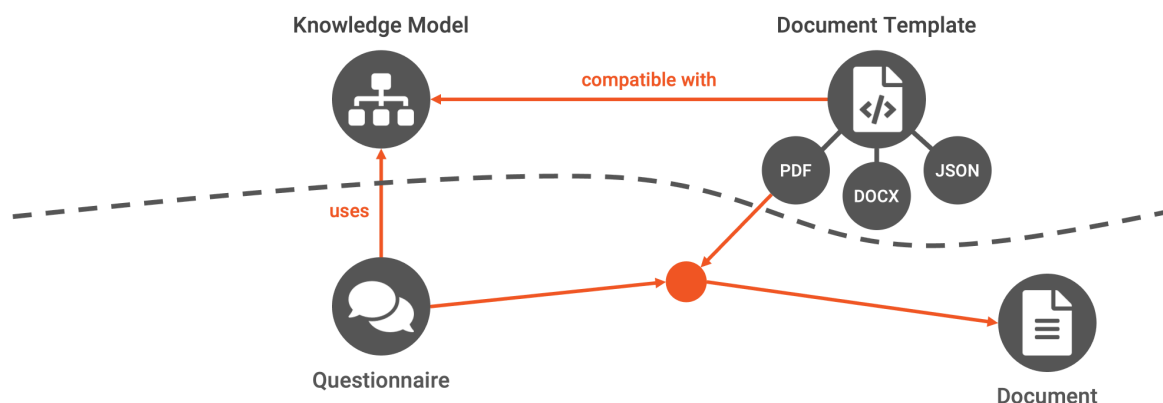
- Le pôle Organisations et Stratégies d'Acteurs (OSA) qui travaille sur les stratégies des entreprises agricoles et agro-alimentaires et les pratiques d'achat des consommateurs, avec un focus particulier sur les transitions bio-économiques et numériques.
- Le pôle Régulations qui analyse et accompagne les régulations publiques, privées et collectives, visant à promouvoir des systèmes agricoles et alimentaires plus durables.
- Le pôle Sécurité Alimentaire et Nutritionnelle Durable (SAND) qui travaille sur les facteurs influençant la consommation alimentaire et l'état nutritionnel des populations, en particulier des plus vulnérables, afin d'améliorer les interventions et d'influencer les politiques publiques en matière de sécurité alimentaire et nutritionnelle durable des populations.

### 3. Présentation de Data Stewardship Wizard

DSW est né de la collaboration en 2016 entre des professeurs et chercheurs et grâce au support de l'organisation intergouvernementale ELIXIR\*. Ce projet a été fondé sur des idées et technologies de plusieurs chercheurs. Il offre une organisation et une abstraction des PGD très puissantes. Il a été développé en utilisant Haskell et Elm, deux langages de programmation fonctionnelle reconnus pour leur efficacité. Dans la suite de ce rapport, je vais vous présenter la structure de cet outil afin que vous puissiez mieux comprendre le travail que j'ai effectué.

DSW utilise plusieurs composants connectés qui permettent de créer un document : Le Knowledge Model, le Questionnaire, le Document Template, et le Document. Voici un diagramme illustrant le lien et la corrélation de ces composants :

#### Data Steward

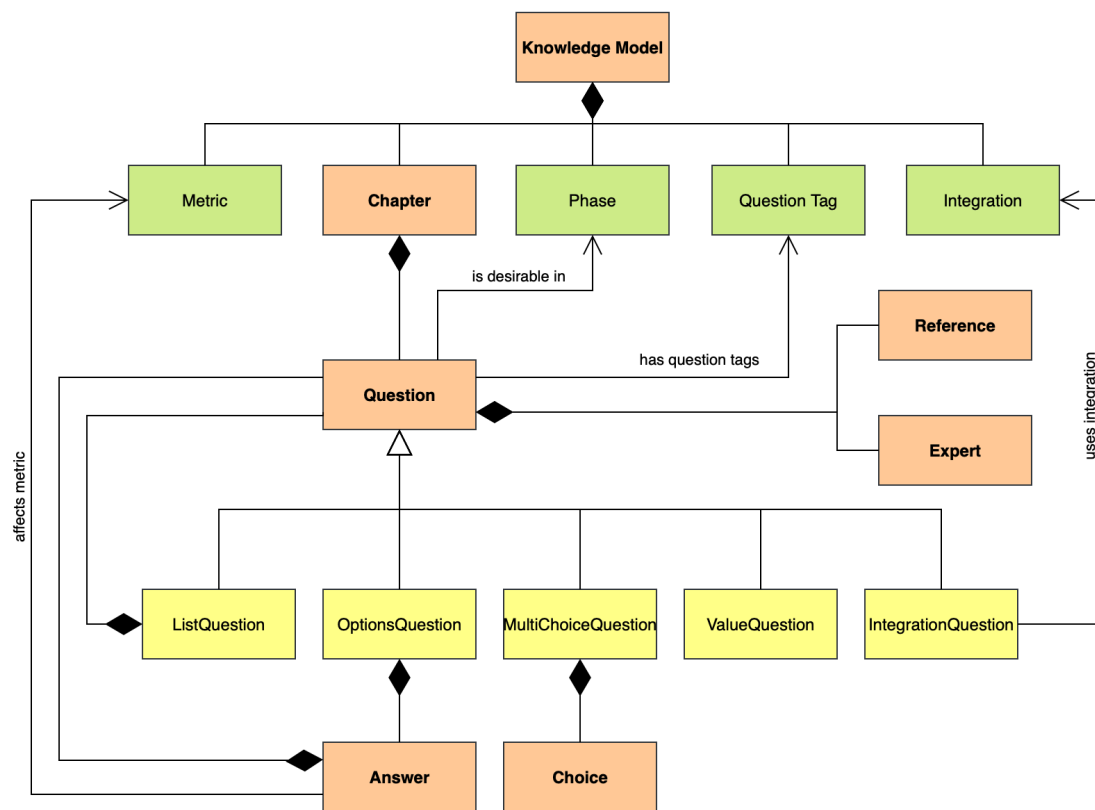


#### Researcher

source : guide de DSW

#### 3.1. Le Knowledge Model (KM)

Le KM ou Knowledge Model est une structure en forme d'arborescence qui agit manière interactive aux utilisateurs en fonction de leurs réponses. Il agit comme modèle/template pour le Questionnaire. C'est sur ce modèle qu'on définira la structure du document et où on abordera les questions auxquelles le responsable de projet devra répondre. Voici un diagramme illustrant la structure d'un KM :



Nous pouvons comparer un KM à une classe abstraite en programmation orientée objet (POO). Elle définit la structure du Questionnaire (PGD) mais ne contient aucune autre donnée.

## 3.2. Le Questionnaire

Dans le cadre de l'UMR MoISA, le Questionnaire est le socle du PGD. Un Questionnaire va se baser sur la structure d'un KM, il faudra répondre aux questions qui figurent dans ce KM. Il sera possible d'exporter ce document en plusieurs formats différents en utilisant le Document Template.

Le Questionnaire est composé de questions que nous exposerons en considérant un exemple appliqué aux données à caractère personnel. Dans ce cas, le PGD est un outil de mise en conformité aux exigences du RGPD. Voici tous les types de questions possibles :

### 3.2.1. Options Question

L'Options Question possède une liste de réponses prédéfinies parmi lesquelles nous pouvons en choisir une. En fonction du choix, il peut y avoir des questions de suivi qui seront elles-mêmes des Options Question, List Question, etc.

Voici une sortie d'écran qui illustre une question de type Options Question :

The screenshot displays a user interface for a questionnaire. It features two questions of the 'Options Question' type. The first question, 'Utilisation d'un jeu de données externes utilisées pour le projet / Appariement de fichiers', has two radio button options: 'a. Oui' (selected) and 'b. Non'. Below the options is a 'Clear answer' button and a timestamp 'Answered 11 days ago by Albert Einstein.'. The second question, 'Caractérisation du jeu de données', includes a 'Test' button and a text input field containing the word 'Test'. It also has a 'Clear answer' button and a timestamp 'Answered 3 minutes ago by Albert Einstein.'. Both questions are marked as 'Desirable: Before Submitting the Proposal'.

Ici, l'utilisateur a répondu oui et doit par conséquent répondre à de nouvelles questions.

### 3.2.2. List Question

La List Question ne comporte pas une simple réponse, mais une liste d'éléments. Chacun de ces éléments possède le même ensemble de sous-questions. Par exemple, dans une List Question, nous demandons au porteur du projet d'identifier l'ensemble contributeur en spécifiant leur nom, leur rôle, etc.

Voici une sortie d'écran qui illustre une question de type Options Question :

## III. Chapitre 3. Données internes

Traitement de données

### ✓ III.1 Jeu de données



☒ Desirable: *Before Submitting the Proposal*

> Test	↓	🗑️	
> Test	↑	↓	🗑️
> Test	↑	🗑️	
+ Add			

<	Previous Chapter Chapitre 2. Le projet	Next Chapter Chapitre 4. Jeu(x) de données externe(s)	>
---	---	--	---

Et il est possible d'ajouter un autre élément à la List Question avec le bouton "Add" :

## III. Chapitre 3. Données internes

Traitement de données

### ✓ III.1 Jeu de données



☒ Desirable: *Before Submitting the Proposal*

> Test	↓	🗑️	
> Test	↑	↓	🗑️
> Test	↑	↓	🗑️
> Autre Test	↑	🗑️	
+ Add			

<	Previous Chapter Chapitre 2. Le projet	Next Chapter Chapitre 4. Jeu(x) de données externe(s)	>
---	---	--	---

Ces éléments contiennent également des questions :



Test

III.1.d.1
Numéro du questionnaire

☒ Desirable: Before Submitting the Proposal

Autre Test

Clear answer

Answered 6 minutes ago by Albert Einstein.

III.1.d.2
Type de données

☒ Desirable: Before Submitting the Proposal

☒ a. Qualitative

☐ b. Quantitative

Clear answer

Answered 6 minutes ago by Albert Einstein.

III.1.d.3
Population / Répondants

Indiquer la population enquêtée: ménage, consommateur, exploitant agricole ...

☒ Desirable: Before Submitting the Proposal

Test

Clear answer

### 3.2.3. Value Question

La Value Question contient un champ de saisie pour notre réponse. Il peut s'agir d'un simple champ de texte ou par exemple d'un sélecteur de date et d'autre type de saisie.

Voici une saisie d'écran illustrant une Value Question qui permet de saisir une date :

1.2
Date

YYYY/MM/DD

☒ Desirable: Before Submitting the Proposal

2023-06-11

June
2023

Sun Mon Tue Wed Thu Fri Sat

28 29 30 31 1 2 3

4 5 6 7 8 9 10

11 12 13 14 15 16 17

18 19 20 21 22 23 24

25 26 27 28 29 30 1

2 3 4 5 6 7 8

### 3.2.4. Multi-choice Question

La Multi-choice Question ressemble fortement à l'Options Question, cependant, nous pouvons choisir plusieurs réponses et il n'y a pas de questions de suivi par rapport à une réponse.

La précision des différents types de question sera utile pour expliquer mes choix dans la partie résultat de ce rapport.

## 3.3. Le Document Template

Le Document Template est un outil qui permet de convertir les données saisies dans un Questionnaire en un format document et lisible. Cela peut inclure des formats plus adaptés pour une utilisation par des machines, tels que JSON, ou des formats plus adaptés à la lecture humaine, tels que DOCX ou PDF. Cette conversion de format peut être utile pour faciliter l'utilisation ou la visualisation des données, en fonction des besoins spécifiques du projet ou de l'utilisateur.

## 3.4. Le Document

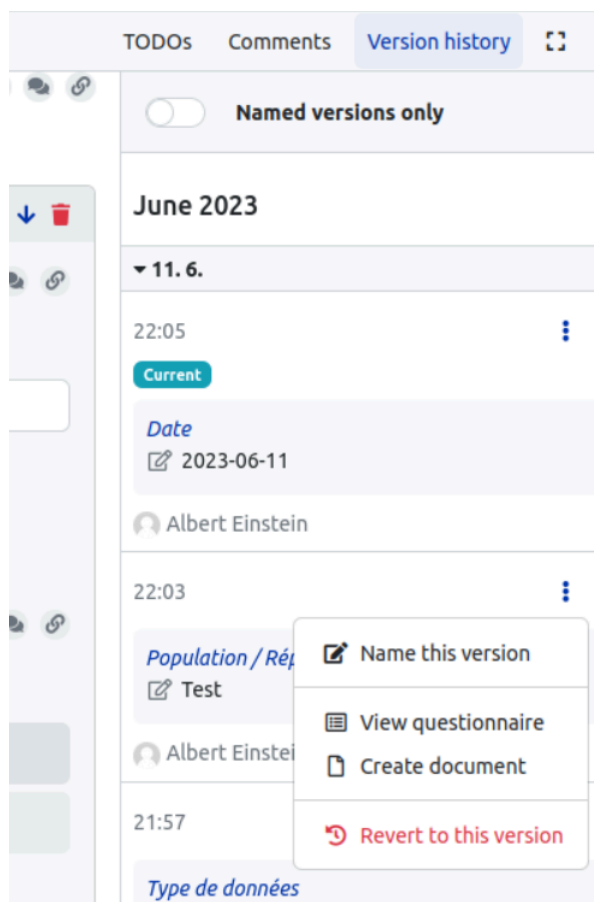
Le document est le résultat de l'exportation d'un des formats du Document Template.

## 3.5. DSW vs DMP OPIDoR

Il est important de comprendre pourquoi l'unité MoISA a choisi d'expérimenter avec cet outil plutôt que d'utiliser DMP OPIDoR.

### 3.5.1. DSW

Tout d'abord, DSW propose un questionnaire évolutif, ce qui signifie que nous pouvons le modifier et le mettre à jour au fil du temps. De plus, il est possible de revenir en arrière sur une ancienne version du questionnaire, ce qui nous permet de corriger des modifications précédentes :



Un autre point fort de DSW est la boucle non bornée. Cela signifie que nous pouvons avoir un nombre indéfini de réponses de questionnaire en questionnaire. Concrètement, nous ne sommes pas limités par une structure linéaire stricte (Cf. [List Question](#)). Cette flexibilité est très utile pour créer des questionnaires complexes et personnalisés qui s'adaptent aux besoins spécifiques d'un projet.

DSW propose également un système de commentaire puissant, similaire à celui de Google Docs ou Word. Cela permet de collaborer efficacement en laissant des commentaires et des suggestions sur le questionnaire. Cela facilite la communication et le travail d'équipe, en permettant de partager certaines idées et d'améliorer collectivement le questionnaire. DSW permet la création de [KM](#) personnalisés. Cette fonctionnalité donne une grande liberté sur la conception et la structure du questionnaire.

DSW est populaire Europe et possède par conséquent une grande ouverture d'esprit. Étant largement utilisé dans la Communauté européenne, cet outil bénéficie de nombreux retours d'expériences et d'une grande communauté d'utilisateurs. Cela garantit une meilleure prise en compte des besoins spécifiques et une évolution continue de l'outil en fonction des meilleures pratiques et des retours des utilisateurs.

Cependant, au sein de l'institut INRAE DSW est uniquement utilisé au sein des unités MoISA et CEE-M, ce qui peut entraîner des problèmes de partage, de

collaboration et de réutilisation des données avec d'autres unités au sein de l'institut. Si ce point mérite d'être souligné, il doit toutefois être relativisé dans la mesure où un PGD est construit par projet et le porteur du projet peut choisir de le saisir, ou pas, sous DSW. Dans le cas où le PGD serait saisi sous DSW, le chercheur peut demander l'ouverture d'un accès pour que les partenaires du projet puissent modifier le PGD saisi.

### 3.5.2. DMP OPIDoR

DMP OPIDoR est fortement utilisé au sein des instituts de recherche français, ce qui facilite la collaboration avec cet outil.

Cependant, cet outil ne permet pas de créer sa propre structure de questionnaire. Les utilisateurs doivent se conformer à une structure prédéfinie proposée par le site, ce qui limite la flexibilité et la personnalisation des PGD en fonction des besoins spécifiques de chaque projet de recherche.

DMP OPIDoR offre lui également un système de commentaire, mais celui-ci ne possède pas un système de résolution comme celui de DSW. Cela signifie qu'il n'est pas possible de savoir si un problème ou une question soulevé(e) dans un commentaire a été résolu ou pris en compte par les utilisateurs. Par ailleurs, il n'est pas possible d'identifier rapidement et simplement les questions soulevées. Ces limitations peuvent entraîner une certaine confusion ou un manque de suivi clair des discussions entre partenaires d'un projet. Enfin, les utilisateurs sont limités aux formats prédéfinis proposés par la plateforme puisqu'il n'est pas possible de personnaliser le document d'exportation du PGD.

Nous pouvons donc constater que DSW propose des fonctionnalités intéressantes. C'est pour cela que l'unité MoISA a choisi d'expérimenter avec cet outil.

## 4. Analyse

### 4.1. Exportation du PGD

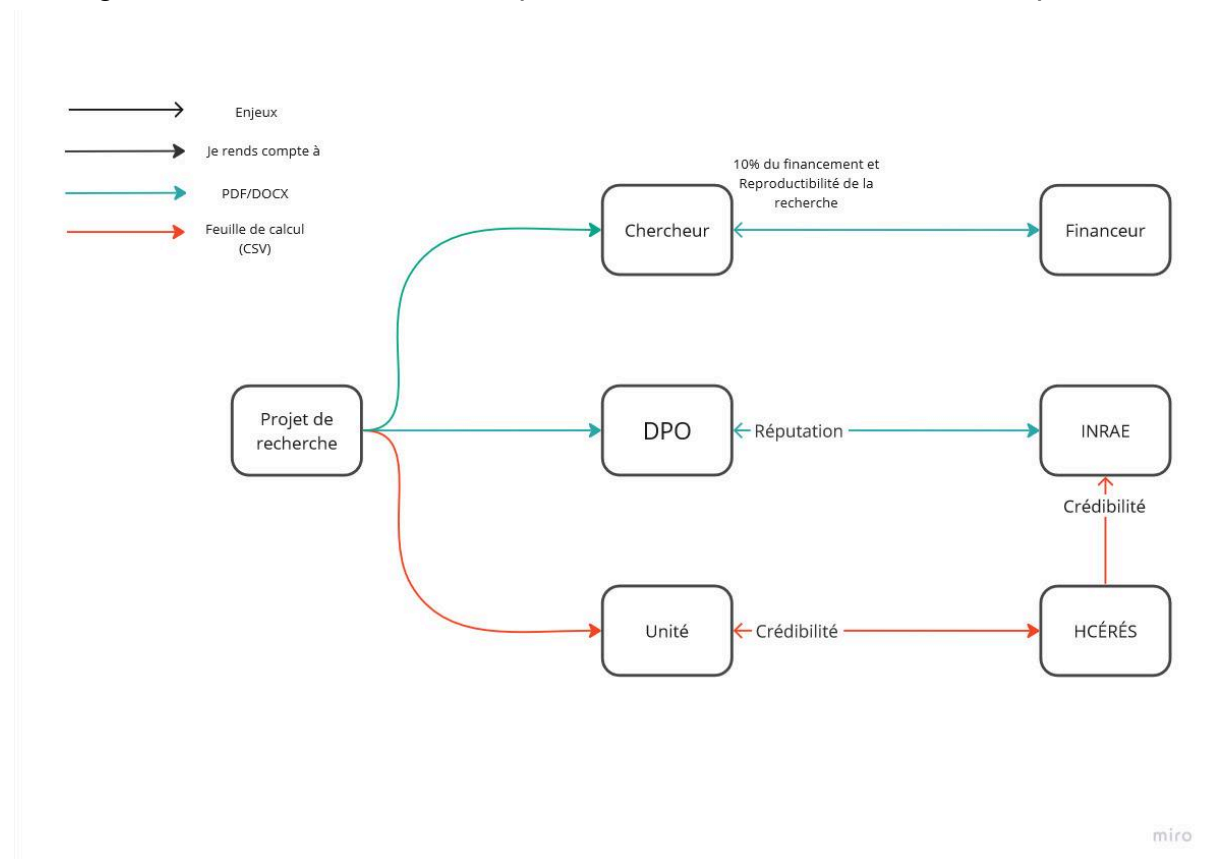
L'UMR MoISA est l'unité pilote chargée de tester la pertinence de mobiliser DSW pour saisir des PGD dans le cadre du processus de mise en conformité des pratiques avec les exigences réglementaires du RGPD. Les chercheurs au sein de cette unité utilisent DSW depuis deux ans. Cependant, notre tutrice, également correspondante-DPO\* de l'unité MoISA, Magali Aubert, a constaté que les fonctionnalités existantes de DSW n'étaient pas pleinement exploitées et qu'il y avait des extensions potentielles à implémenter. Elles pourraient faciliter la rédaction des PGD et faciliter la valorisation collective des PGD saisis individuellement.

Une fonctionnalité intéressante de DSW est l'exportation automatique des [Questionnaires](#) dans différents formats (PDF, DOCX, LaTeX, JSON, etc.) grâce au [Document Template](#). Cependant, le rendu de l'exportation du Questionnaire était très basique et ne comportait pas de mise en page, ce qui le rendait difficile à lire pour les utilisateurs. Notre tutrice devait donc copier toutes les informations figurant dans le Questionnaire DSW vers un document DOCX\*. Cette démarche peut s'avérer extrêmement chronophage et répétitif.

Notre première mission a donc été de développer des formats de mise en page plus lisibles en utilisant un outil fourni par DSW appelé TDK. Ce dernier nous permet de créer un Document Template localement et de programmer la mise en page des différents formats en utilisant le langage de templating Jinja2.

Mme Aubert avait également besoin d'exporter les PGD en CSV\* pour effectuer des analyses statistiques avec Stata\*. J'ai choisi de me concentrer sur la partie CSV de l'exportation, alors que mon collègue de stage, Jocelin Sanchez, s'est concentré sur l'exportation pour les documents de mise en forme (DOCX, PDF).

Le diagramme suivant illustre les implications des différents formats d'export :



Les chercheurs utiliseront principalement le format **PDF/DOCX** pour prouver la reproductibilité de leurs travaux aux financeurs. Un travail de recherche est dit reproductible si toutes les informations concernant ce travail sont rendues disponibles de telle sorte que n'importe quel chercheur indépendant peut reproduire les résultats. C'est un critère essentiel pour les financeurs. En utilisant ce format, les chercheurs pourront également prouver l'originalité des données générées dans le cadre du projet de recherche. Tout l'enjeu des financeurs et de la Science Ouverte est de renforcer les échanges entre les recherches. La réalisation du PGD renforce la fiabilité des résultats et l'acronyme du principe FAIR et peut par ailleurs avoir un impact sur le financement du projet de recherche. Dans le cas où un PGD ne serait pas fourni lors de la réalisation du projet, 10 % du financement pourrait être retiré.

De plus, la DPO d'INRAE a besoin d'un format standardisé pour valider un projet de recherche et s'assurer qu'il respecte les exigences réglementaires du RGPD. Cette validation par la DPO de cette mise en conformité renvoie directement à la réputation d'INRAE. En respectant les exigences du RGPD, INRAE et la DPO renforcent leur réputation et leur engagement envers la transparence et l'intégrité scientifique.

Le format CSV et les analyses statistiques qui vont être effectuées grâce à ce format permettront à Mme Aubert de prouver lors des évaluations HCERES de son unité que les données de terrain collectées sont traitées conformément aux exigences réglementaires du RGPD. Plus globalement, ceci donnera plus de crédibilité à l'unité et ainsi à INRAE.

## 4.2. Stockage du PGD sous Nextcloud

Du fait d'une réflexion sur le choix de l'interface entre DSW et Opidor, notre deuxième mission était de voir si nous pouvions proposer aux utilisateurs de DSW d'exporter leurs PGD vers OPIDoR. Celle-ci visait la migration des PGD saisis sous DSW vers OPIDoR dans le cas où DSW n'arriverait pas à s'implanter à INRAE. Cependant, cette mission a été modifiée en cours de projet après une réunion avec le RDO\* du CIRM\*, Jonathan Mineau-Cesari, également membre de la cellule DipSO de INRAE. Selon lui, il n'était pas pertinent d'établir un lien entre DSW et DMP OPIDoR. En effet, l'interopérabilité DSW-OPIDOR n'est pas encore possible du fait d'un manque d'accès aux fonctionnalités d'OPIDOR. Cette expérience, même si elle n'a pas pu être menée à terme, nous a permis de découvrir une interface différente par rapport à celle de DSW. Grâce à cela, il a été plus simple de nous approprier certaines de ses spécificités comparativement à l'existant. Cette prise de recul sur cet outil nous a permis de gagner en compétence sur DSW en l'appréhendant sous un autre angle. On peut mieux connaître les outils à disposition des chercheurs pour leur saisie de PGD.

L'objectif de cette mission a été redéfini. Il n'était plus question de permettre d'exporter vers OPIDoR, mais de proposer une manière efficace et automatisée de stocker les PGD et tout autre fichier associé dans le cloud d'INRAE : Nextcloud. Ceci permettra d'une part de centraliser les PGD créés sous DSW sur un même endroit et d'autre part de tracer l'ensemble des PGD saisis au niveau de l'unité. Un des enjeux est également d'implémenter de la gestion des versions d'un PGD sur Nextcloud. Cela pourra renforcer cette notion de versionning au cas où le PGD saisi évolue et se trouve modifié au cours du projet. Je reviendrai sur Nextcloud ultérieurement (Cf. [Page 22](#)).

Actuellement, INRAE possède une instance de DSW sur le lien <https://dsw.ecosocio.inrae.fr>. Cette version de DSW n'est pas à jour, mais nous avons communiqué avec un des ingénieurs impliqués dans DSW. Il sera possible non seulement de mettre à jour l'instance de DSW, mais aussi de rajouter des modules (services dans Docker) que nous aurons développés au long du stage. Mme Aubert a déjà créé un [KM](#) sur cette instance DSW, qui a été validé par l'ANR\*, le CIRAD\*, INRAE et la DPO d'INRAE. Ce KM va nous être très utile lors de la réalisation des missions, car nous baserons notre code sur sa structure.

Le travail que nous effectuerons aura un impact significatif sur les unités MoISA et CEEM d'INRAE, pour lesquels les PGD sont saisis sous l'interface DSW. Grâce aux nouvelles fonctionnalités que nous implémenterons, les chercheurs pourront créer leurs PGD de manière plus efficace et les exporter dans des formats lisibles (DOCX, PDF, etc.) par la DPO et les financeurs. De plus, le système de stockage que nous mettrons en place permettra à Mme Aubert de gérer et de valider les PGD de manière plus optimisée.

Notre objectif final est donc de promouvoir DSW au sein d'INRAE et d'élargir son utilisation. En développant ces nouvelles fonctionnalités et en améliorant l'instance actuelle de DSW, nous visons à démontrer les avantages et l'efficacité de l'outil, incitant ainsi davantage de chercheurs et d'unités INRAE à adopter DSW pour la rédaction de leurs PGD. Au final, notre travail contribuera à renforcer la présence de DSW au sein d'INRAE et à faciliter la gestion des PGD pour les chercheurs et les responsables de projet tels que Mme Aubert.

## 4.3. Nextcloud

Nextcloud joue un rôle central dans la réalisation de la deuxième mission. Je vais vous présenter les spécificités de cet outil et vous expliquer pourquoi nous avons choisi de l'utiliser pour ce stage.

Nextcloud est une solution open source conçue pour l'hébergement de fichiers, visant à faciliter l'accès et le partage de données stockées dans différents entrepôts d'INRAE. Cette solution est hébergée dans les datacenters d'INRAE, situés à Toulouse et à Paris. Elle offre aux agents de cet institut un accès simple et direct à des espaces de données en ligne. Le portail Nextcloud est accessible à tous les agents d'INRAE, favorisant la collaboration et le partage d'informations au sein de l'organisation. Nextcloud propose beaucoup de fonctionnalités telles que son [API\\*](#) [WebDAV](#), le stockage sécurisé des fichiers, la gestion des autorisations d'accès et la synchronisation des données.

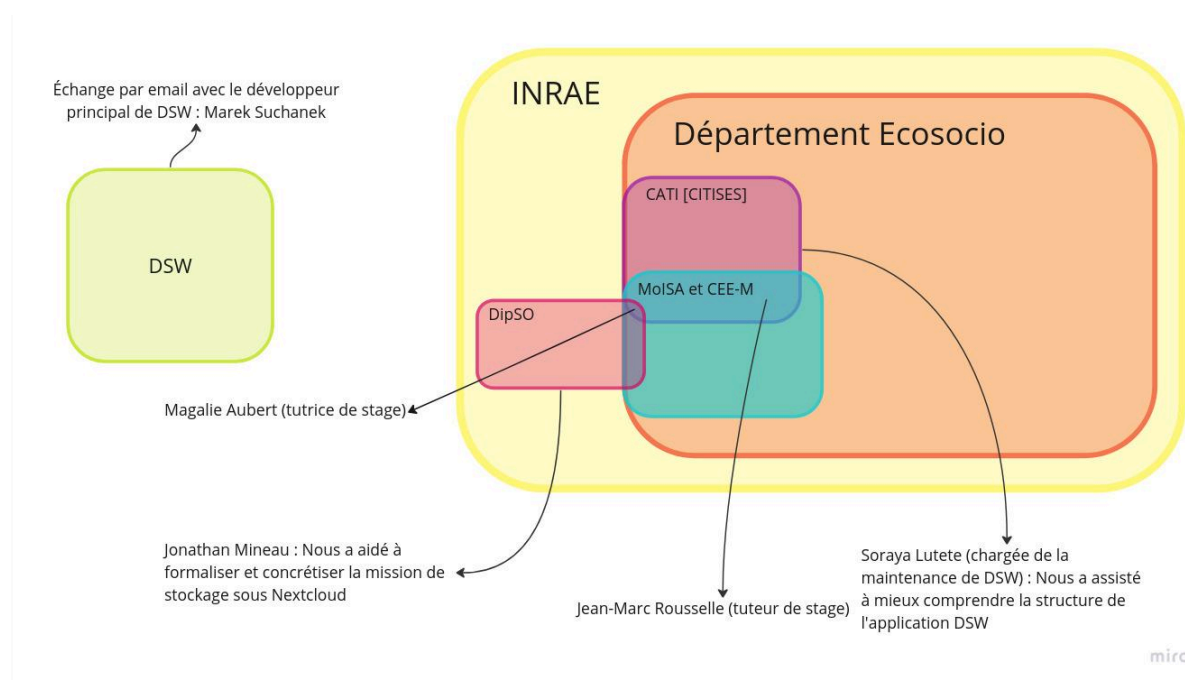
Il a été décidé que nous utilisons Nextcloud pour notre stage, car il est largement adopté au sein d'INRAE et qu'il peut être accessible, par des identifiants sécurisés, à des partenaires de projets. Cet outil est en alignement avec les principes FAIR parce qu'il simplifie la recherche et l'accès aux données grâce à son interface intuitive et son stockage centralisé. Nextcloud encourage l'interopérabilité avec d'autres systèmes grâce à ces API et permet donc de réutiliser les données stockées d'une manière optimale et efficace.



En ce qui concerne la conformité au RGPD, Nextcloud et INRAE mettent en place des mesures de sécurité et de confidentialités robustes pour garantir la protection des données à caractère personnel. L'outil propose des fonctionnalités telles que la gestion des autorisations d'accès, le chiffrement des données, ainsi que des fonctionnalités de suivi des activités pour assurer la transparence des actions effectuées sur les données.

## 4.4. Portée du stage

Pendant ce stage, j'ai eu l'opportunité de collaborer avec plusieurs chercheurs, élargissant ainsi la portée de mon stage. Voici un diagramme illustrant cette portée :



Le **CATI CITISES** est une structure de service informatique et de statistiques spécialisée dans le traitement de l'information économique et sociale. Il est piloté par le département EcoSocio et collabore avec d'autres CATI au sein du département. Le CATI se compose de quatre pôles fonctionnels qui travaillent en étroite collaboration.

La **Direction pour la Science Ouverte (DipSO)** est une entité au sein d'INRAE qui se consacre à répondre aux enjeux liés à l'ouverture de la recherche dans un contexte numérique en constante évolution, et face aux attentes croissantes de la société. Son principal objectif est d'apporter un soutien aux chercheurs en matière de gestion, de partage et de valorisation de leurs données de recherche, tout en promouvant les bonnes pratiques de la science ouverte.

**Magali Aubert**, membre de CATI CITISES et de DipSO, ainsi que l'unité MoISA du département EcoSocio, a joué un rôle très important dans la définition des missions et nous a assisté tout au long de ce stage.

**Jean-Marc Rouselle**, a également contribué à la définition des missions de mon stage. Il est responsable du CATI CITISES et membre de l'unité CEE-M du département EcoSocio.

**Soraya Lutete**, en tant que responsable de la maintenance de DSW pour le département EcoSocio et membre de CATI CITISES, a pu nous introduire à la structure de DSW.

**Jonathan Mineau**, membre de DipSO, a été d'une grande aide pour formaliser et concrétiser la mission de stockage sous Nextcloud.

Nous avons de plus communiqué avec le développeur principal de DSW : **Marek Suchanek**.

La collaboration avec ces chercheurs et ces développeurs a considérablement agrandi la portée de notre stage. Elle peut contribuer à promouvoir l'utilisation de DSW au sein d'INRAE. Mais aussi à créer de nouveaux liens au niveau européen. Il serait question de traduire le [Knowledge Model](#) existant de MoISA en anglais pour le partager à d'autres chercheurs à l'international voulant rédiger un PGD.

## 5. Rapport technique

### 5.1. Apprentissage et consolidation de DSW

Avant toute chose, il est nécessaire de se familiariser avec DSW pour se faciliter le développement des missions confiées. Heureusement, cet outil possède une documentation<sup>[1]</sup> détaillée qui nous a été d'une grande utilité et a servi lors de la réalisation des missions. De plus, il est possible de déployer une instance locale de DSW avec à un dépôt git<sup>[7]</sup> fourni par DSW. Cette approche nous offre une plus grande liberté pour effectuer nos tests et développer des extensions, sans avoir d'impact sur la version existante de DSW au sein d'INRAE.

#### 5.1.1. Structure de DSW sous Docker

**Docker** est un logiciel open source qui permet de déployer et de gérer des applications dans des conteneurs. Un conteneur Docker est une unité légère et isolée du système capable d'exécuter une application. Docker facilite la création de conteneurs autonomes et portables, qui peuvent être exécutés sur différentes plateformes, qu'il s'agisse de machines locales, de serveur ou de services cloud.

Docker Compose, est un outil qui simplifie le déploiement et la gestion d'applications utilisant plusieurs conteneurs. Il permet de décrire l'architecture d'une application dans un fichier YML, en spécifiant les différents services, leurs dépendances et leurs configurations. Docker Compose permet de lancer facilement plusieurs conteneurs de Docker en utilisant une seule commande : **docker compose up (-d)**

Grâce à Docker Compose, il est possible de définir des réseaux pour que les conteneurs puissent communiquer. Il est également possible de configurer les volumes de données partagées et de définir des variables d'environnement. Cela simplifie considérablement la gestion des applications complexes composées de plusieurs conteneurs interconnectés tels que DSW.

Maintenant, abordons la structure et la configuration du fichier docker-compose fourni par DSW. Vous pouvez consulter ce document en annexes (Cf. [docker-compose.yml](#)).

## dsw-server

```
services:
  dsw-server:
    image: datastewardshipwizard/wizard-server:3.23.0
    restart: always
    ports:
      - 127.0.0.1:3000:3000
    depends_on:
      - postgres
      - minio
    volumes:
      - ./config/application.yml:/app/config/application.yml:ro
    extra_hosts:
      - host.docker.internal:host-gateway
```

Le service `dsw-server` représente la partie backend de l'application. Le backend en développement web est la partie invisible qui gère les fonctionnalités et les données du site web. Ce service a été entièrement développé avec le langage Haskell. Voici un lien vers le code source de ce service :

<https://github.com/ds-wizard/engine-backend>,

`dsw-server` propose une API REST, vous pouvez consulter sa documentation sur le lien suivant : <https://api-docs.ds-wizard.org/>. Cette documentation a été essentielle pour comprendre le fonctionnement de DSW et nous a permis d'évaluer la faisabilité des missions du stage. Une fois que l'application tourne sur Docker, ce service est accessible sur l'adresse <http://localhost:3000>. `dsw-server` dépend d'autres services dans le fichier docker-compose que je vais vous introduire bientôt.

## dsw-client

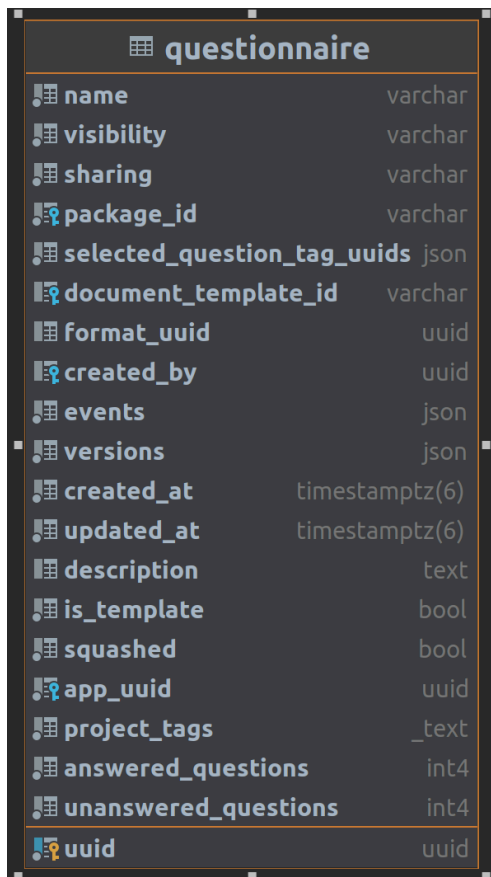
```
dsw-client:
  image: datastewardshipwizard/wizard-client:3.23.0
  restart: always
  ports:
    - 127.0.0.1:8080:8080
  environment:
    API_URL: http://localhost:3000
```

Le service `dsw-client` représente la partie frontend de DSW. Le frontend en développement web est la partie visible et interactive du site web avec laquelle les utilisateurs interagissent directement. Cette partie de l'application a été développée utilisant le langage Elm. Ce service est actif sur l'adresse <http://localhost:8080>. Il est très dépendant du service backend comme c'est celui qui fournit toutes les informations à travers son API REST.

## postgres

```
postgres:
  image: postgres:15.2
  restart: always
  # (!! ) Expose only for debugging locally
  # ports:
  #   - 127.0.0.1:5432:5432
  # (!! ) Change default password
  environment:
    POSTGRES_DB: engine-wizard
    POSTGRES_USER: postgres
    POSTGRES_PASSWORD: postgres
  # (!! ) Mount for persistent data
  # volumes:
  #   - db-data:/var/lib/postgresql/data
  # OR
  #   - ./db-data/data:/var/lib/postgresql/data
```

Voici la base de données de l'application qui utilise le SGBD\* PostgreSQL. Elle est accessible sur le port localhost:5432. Je peux visualiser les données grâce à un IDE\* comme DataGrip.



questionnaire	
name	varchar
visibility	varchar
sharing	varchar
package_id	varchar
selected_question_tag_uuids	json
document_template_id	varchar
format_uuid	uuid
created_by	uuid
events	json
versions	json
created_at	timestamp(6)
updated_at	timestamp(6)
description	text
is_template	bool
squashed	bool
app_uuid	uuid
project_tags	_text
answered_questions	int4
unanswered_questions	int4
uuid	uuid

Voici la table du [Questionnaire](#) de DSW. Il y a toutes les informations d'un PGD dans une ligne de cette table. Par exemple, l'attribut `events` de cette table contient les données des réponses, de l'historique etc.

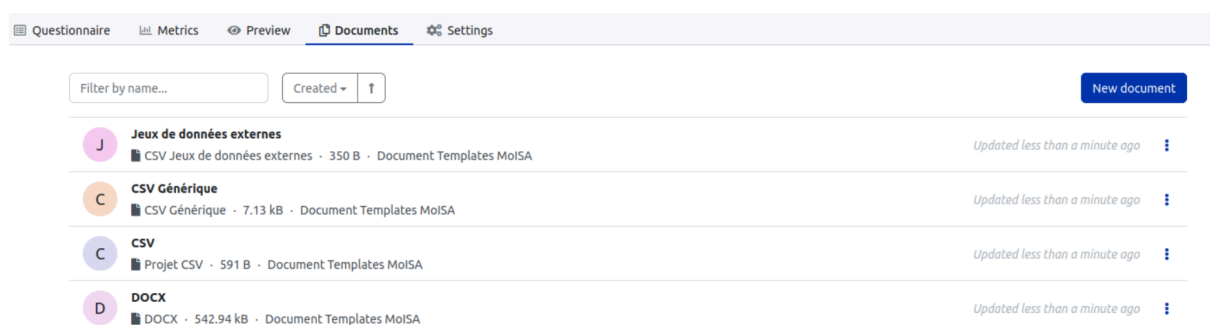
Le backend engine de DSW va traiter les données stockées dans cette base de données et retourner une réponse JSON qui sera traitable et "machine-actionnable". En effet, ceci est le rôle même d'une API REST. Celle-ci agit comme intermédiaire entre le client (`dsw-client`) et le serveur (`postgres` et `dsw-server`).

Elle recevra des requêtes du client, les transmettra au serveur, puis récupérera les réponses données par ce dernier pour les renvoyer au client.

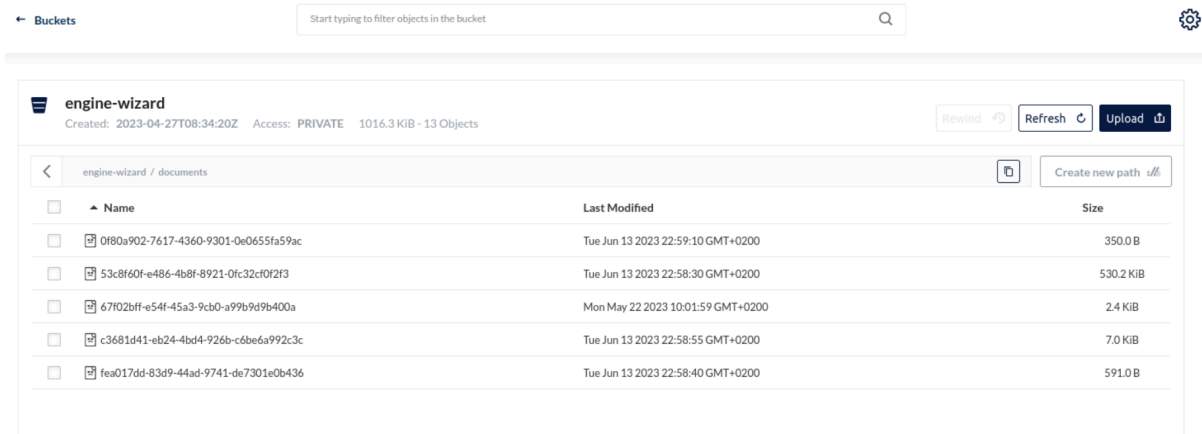
## minio

```
minio:
  image: minio/minio:RELEASE.2022-08-02T23-59-16Z
  restart: always
  command: server /data --console-address ":9001"
  # (!! ) Expose only for debugging locally, externally use HTTPS proxy (see MinIO docs)
  ports:
    - 9000:9000
    - 9001:9001
  # (!! ) Change default password
  environment:
    MINIO_ROOT_USER: minio
    MINIO_ROOT_PASSWORD: minioPassword
  # (!! ) Mount and backup for persistent data
  # volumes:
  #   - s3-data:/data
  # OR
  #   - ./s3-data/data:/data
```

MinIO est un système de stockage d'objets open-source conçu pour être simple, évolutif et hautement performant. Il est compatible avec l'API de stockage d'objets S3 d'Amazon. MinIO est conçu pour être déployé en tant que service de stockage autonome, que ce soit sur une infrastructure locale ou sur le cloud. Dans l'infrastructure DSW, ce système de stockage est utilisé pour stocker les [Documents](#) exportés.



Les fichiers affichés sur cette capture d'écran sont stockés sur MinIO. Si nous nous connectons sur l'interface de MinIO sur l'adresse <http://localhost:9000>. Nous pouvons retrouver des fichiers nommés avec des UUIDs\*. Ceux-ci correspondent aux documents listés dans la capture d'écran ci-dessus.



## docworker

```
docworker:
  image: datastewardshipwizard/document-worker:3.23.0
  restart: always
  depends_on:
    - postgres
    - minio
    - dsw-server
  volumes:
    - ./config/application.yml:/app/config/application.yml:ro
  extra_hosts:
    - host.docker.internal:host-gateway
```

Ce service nous permet d'exporter un [Questionnaire](#) en un format proposé par la [Document Template](#) que nous allons développer.

Maintenant que vous avez la structure de DSW en tête, il est temps pour moi de m'approprier l'API de DSW. Il suffit d'avoir installé docker sur votre système, d'ouvrir un terminal sur la racine du projet et taper la commande suivante : **docker compose up -d**.

```
[+] Building 0.0s (0/0)
[+] Running 8/8
✓ Network dsw-deployment-example_default          Created
✓ Container minio                                  Started
✓ Container dsw-deployment-example-dsw-client-1   Started
✓ Container dsw-deployment-example-postgres-1     Started
✓ Container dsw-server                             Started
✓ Container dsw-deployment-example-nextcloud-submitter-1 Started
✓ Container dsw-deployment-example-mailer-1       Started
✓ Container dsw-deployment-example-docworker-1    Started
```

Ça fonctionne !

### 5.1.2. Test de l'API DSW

l'API de DSW va être un élément essentiel pour la réalisation de la deuxième mission de mon stage. Nous avons vu que le point d'accès à l'API de DSW est <http://localhost:3000>. Un outil très utile pour tester l'API est [Postman](#). DSW fournit également une documentation de l'API<sup>[13]</sup> en utilisant [SwaggerUI](#) qui permet de la visualiser et d'interagir à travers une interface.

Par exemple, la requête <http://localhost:3000/tokens> permet de récupérer le token d'authentification. En développement web, un token est une chaîne de caractères utilisée pour l'authentification et l'autorisation des utilisateurs. Il représente l'identité d'un utilisateur ou d'une application et est utilisé pour vérifier l'authenticité des demandes et l'accès aux ressources protégées.

Pour cette requête, il faut donc donner son identifiant et mot de passe en format JSON dans le corps de la requête :

```
POST /tokens HTTP/1.1
Host: localhost:3000
Content-Type: application/json
Content-Length: 92

{
  "code": null,
  "email": "albert.einstein@example.com",
  "password": "password"
}
```

Ceci retourne un token qui permet d'effectuer des requêtes à l'API nécessitant une authentification :

```
{
  "token": "eyJhbGciOiJSUzI1NiJ9..",
  "type": "UserToken"
}
```

Ce [lien](#) vous permettra d'accéder à toutes les requêtes que j'ai effectuées avec Postman pour tester l'API de DSW.



## 5.2. Développement du format CSV

Passons maintenant au développement de la [Document Template](#) proposant un format d'export de type CSV. La documentation de DSW<sup>[1]</sup> possède une page dédiée au développement de ses propres Document Templates. Cette documentation indique qu'il faut utiliser un outil appelé DSW-TDK pour les développer localement

### 5.2.1. Le format CSV

Le CSV (Comma-Separated Values) est un format de fichier utilisé pour stocker des données tabulaires. Les valeurs des colonnes sont séparées par des virgules. Par exemple :

```
Nom,Prénom,Age,Ville  
Dupont,John,30,Paris  
Smith,Jane,25,New York
```

Le caractère de délimitation peut varier (virgule, point-virgule, etc.) et les valeurs peuvent être entourées de guillemets pour échapper le texte à l'intérieur. C'est un format simple et couramment utilisé dans divers domaines comme les analyses statistiques.

C'est donc ce type de données que je dois réussir à générer à partir d'un [Questionnaire](#) en utilisant DSW-TDK

### 5.2.2. DSW-TDK

Voici le lien vers le dépôt GitHub de DSW-TDK : <https://github.com/ds-wizard/engine-tools/tree/develop/packages/dsw-tdk>

Le seul prérequis pour utiliser DSW-TDK est d'avoir python3 installer sur son système. Pour l'installer, il suffit de taper : `pip install dsw-tdk`. La commande `dsw tdk new .` nous permet de créer un nouveau projet :

```
[~/Development]
theodore dsw-tdk new _
Template name: Test
Organization ID: Test
Template ID [test]: test
Version [0.1.0]:
Description [My custom template]: Test
License [CC0]: CC0
=====
Do you want to add a format? [Y/n]: Y
Format name [HTML]: CSV
File extension [csv]: csv
Content type [text/csv]:
Jinja2 filename [src/template.csv.j2]:
=====
Do you want to add yet another format? [y/N]: N
SUCCESS: Template project created: /home/theodore/Development
```

Voici ce que cette commande a généré :

```
theodore 1
total 24K
drwxrwxr-x 3 theodore theodore 4,0K juin 14 00:51 .
drwxrwxr-x 9 theodore theodore 4,0K juin 14 00:51 ..
-rw-rw-r-- 1 theodore theodore 94 avril 25 09:49 .env
-rw-rw-r-- 1 theodore theodore 158 juin 14 00:49 README.md
drwxrwxr-x 2 theodore theodore 4,0K juin 14 00:49 src
-rw-rw-r-- 1 theodore theodore 980 juin 14 00:49 template.json
```

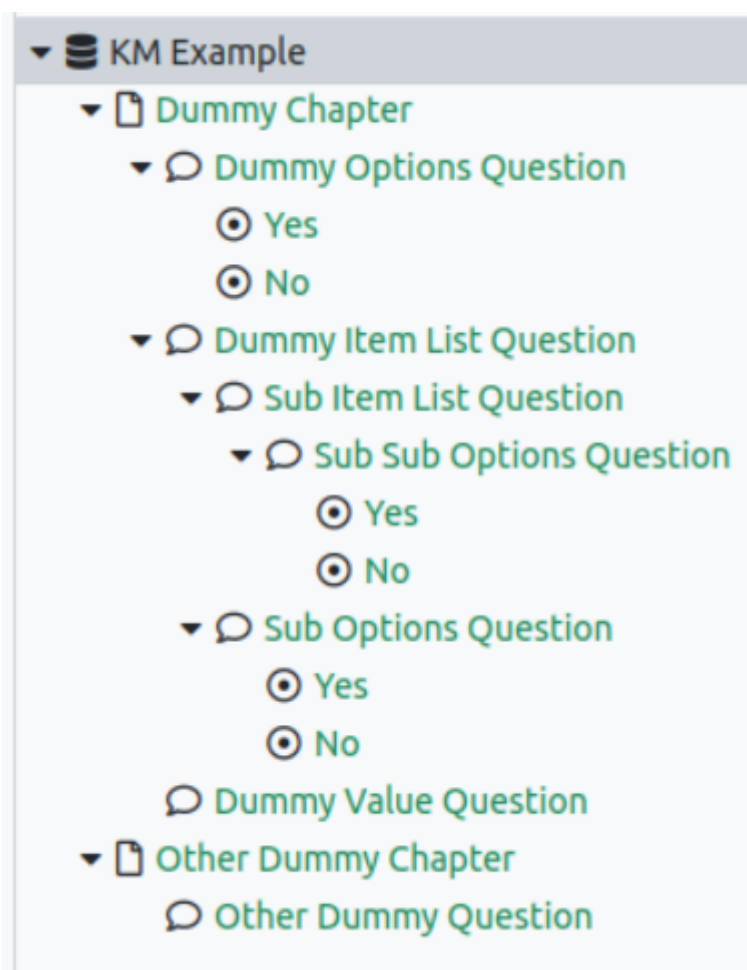
- Le dossier **src** contient les fichiers d'extension jinja2. C'est ici que nous allons développer les différents formats.
- Le fichier **template.json** contient toutes les informations nécessaires pour la construction du Document Template telles que sa version, son nom, une liste des différents formats avec référence sur un fichier jinja2 qui définit le rendu de ce format.

Jinja2 est un moteur de templates Python utilisé pour la génération de contenu et document. Il facilite la séparation de la logique métier et des "vues". La syntaxe de jinja2 est identique à celle de twig pour PHP.

### 5.2.3. Version générique du format CSV

La première version du format que je vais développer est "Générique", c'est-à-dire qu'elle s'adapte à tous les [KM](#). Elle s'adapte aussi à celui créé par Mme Aubert. J'ai choisi de créer un KM simple qui m'aidera à illustrer la problématique de cette mission.

Voici la structure du KM d'exemple :



Avec cette capture d'écran, vous pouvez constater qu'il y a une arborescence imprédictible. Dans un KM, une question peut recueillir une liste non bornée de réponses. En théorie, cette récurrence peut être infinie. J'ai compris qu'il fallait développer des méthodes récursives permettant de rentrer dans cette arborescence.

Sachant qu'uniquement les [Options Questions](#) et [List Questions](#) peuvent contenir une "sous-question", je vais baser mes conditions d'arrêt sur cette information. Pour le développement avec Jinja2, DSW-TDK me fournit une variable de type dict (Python) contenant toutes les informations du [Questionnaire](#) concerné par l'exportation. Cette variable nommée `ctx` est très lourde (plus de 10000 valeurs).

Voici une capture d'écran de sa structure :

```

1  {
2  'config': {"clientId": 'http://localhost:8080'...},
5  'createdAt': '2023-05-05T12:57:12.682834558Z',
6  'createdBy': {"affiliation": 'My University'...},
39 'documentTemplateMetamodelVersion': 11,
40 'knowledgeModel': {"uuid": 'a5da388f-76d6-4337-990c-c85f8c5ea53b'...},
7456 'organization': {"name": 'My Organization'...},
7462 'package': {"id": '00001:pgd:1.0.1'...},
7475 'phaseUuid': 'b101f2d0-2476-452d-aa8d-95a41a02b52c',
7476 'questionnaireDescription': None,
7477 'questionnaireName': 'myProject2',
7478 'questionnaireProjectTags': [],
7479 'questionnaireReplies': {...},
9789 'questionnaireUuid': '9427eec4-821a-4fbc-8ef5-f0248d9010b1',
9790 'questionnaireVersion': None,
9791 'questionnaireVersions': [],
9792 'report': {"createdAt": '2023-05-05T12:57:12.686446073Z'...},
9878 'updatedAt': '2023-05-05T12:57:12.682834558Z',
9879 'uuid': '2ed7f731-3d11-49e9-851c-3b47491c30af',
9880 'extras': {}
9881 }

```

Les deux valeurs importantes dans ce dictionnaire sont 'knowledgeModel' et 'questionnaireReplies'. Elles contiennent respectivement la structure du KM utilisé par le Questionnaire et les réponses du Questionnaire.

Dans un fichier que j'ai nommé **src/generic-template.csv.jinja2**. J'instancie deux variables qui stockeront ces valeurs :

```

{%- set km = ctx.knowledgeModel -%}
{%- set repliesMap = ctx.questionnaireReplies -%}

```

Je peux déclarer les fonctions qui vont traverser récursivement le Questionnaire, voici une partie d'une des méthodes permettant d'afficher le "tag" de la question :

```
{%- macro printQuestionTag(replyPathUuids, question, questionString) -%}
  {%- if (replyPathUuids | reply_path) in repliesMap -%}
    {%- if question.questionType == "ListQuestion" -%}

      {%- set reply = repliesMap[replyPathUuids | reply_path].value -%}
      {%- set k = namespace(value = 0) -%}

      {%- for questionUuid in reply.value -%}
        {%- set i = namespace(value = 1) -%}
        {%- for subQuestionUuid in question.itemTemplateQuestionUuids -%}
          {%- set subQuestion = km.entities.questions[subQuestionUuid] -%}
          {{- replyPathUuids.append(questionUuid) or "" -}}
          {{- replyPathUuids.append(subQuestionUuid) or "" -}}

          {{- printQuestionTag(replyPathUuids, subQuestion, questionString + "_" +
k.value | of_alphabet + "_" + i.value | string) -}}

          {{- replyPathUuids.remove(questionUuid) or "" -}}
          {{- replyPathUuids.remove(subQuestionUuid) or "" -}}

          {%- set i.value = i.value + 1 -%}
        {%- endfor -%}
        {%- set k.value = k.value + 1 -%}
      {%- endfor -%}
    ...
  {%- else -%}
    {{- "\"" + questionString + "\", " -}}
  {%- endif -%}
{%- endmacro -%}
```

Ici, je vérifie si la question donnée en paramètre est de type "[ListQuestion](#)". Si celle-ci possède des "sous-question", je traverse chaque "sous-question" avec une boucle **for**. Ensuite, j'appelle cette même fonction sur cette "sous-question" au cas où celle-ci est elle-même une List Question ou une Options Question. Si la question passée en paramètre n'appartient pas un de ces types, la méthode affiche le tag de la question. Je ne vais pas rentrer davantage dans le code, mais le principe est le même pour toutes les autres méthodes.

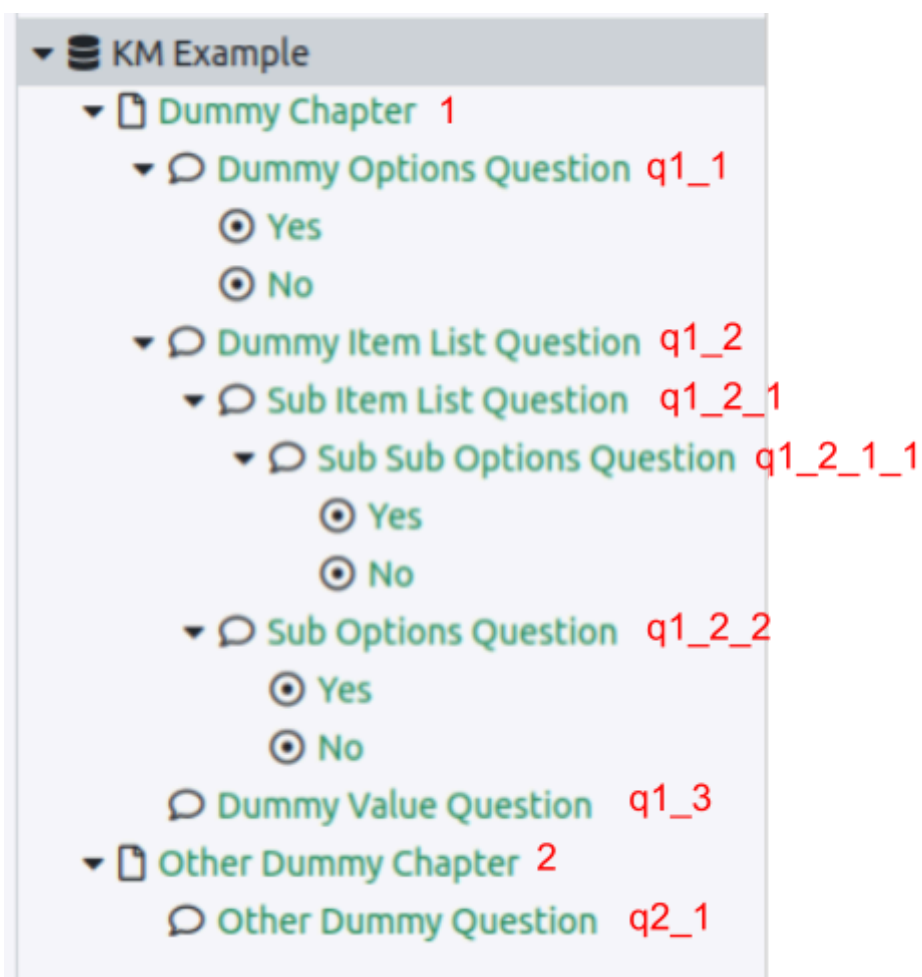
Voici le résultat en format tableur d'un questionnaire se basant sur la structure du KM d'exemple :

q1_1	q1_2_a_1_a_1	q1_2_a_2	q1_3	q2_1
Yes	No	No	Dummy Answer	Test

### Le tag de la question      La valeur de la question

Le tag correspond à une numérotation de la question. Ceci rend le format CSV plus lisible et simple à comprendre en simplifiant également le traitement de ce format sous Stata. Voici un diagramme illustrant la logique des tags :

sn



Comme une [List Question](#) peut contenir plusieurs réponse, il a fallu rajouter des lettres permettant de différencier chaque réponse

q1_1	q1_2_a_1_a_1	q1_2_a_1_b_1	q1_2_a_2	q1_2_b_1_a_1	q1_2_b_2	q1_3
Yes	Yes	No	No	No	No	Dummy Answer

Pour savoir à quoi correspond chaque tag de question, j'ai développé un glossaire qui s'adapte au [Questionnaire](#). En voici un qui se base sur le Questionnaire ci-dessus :

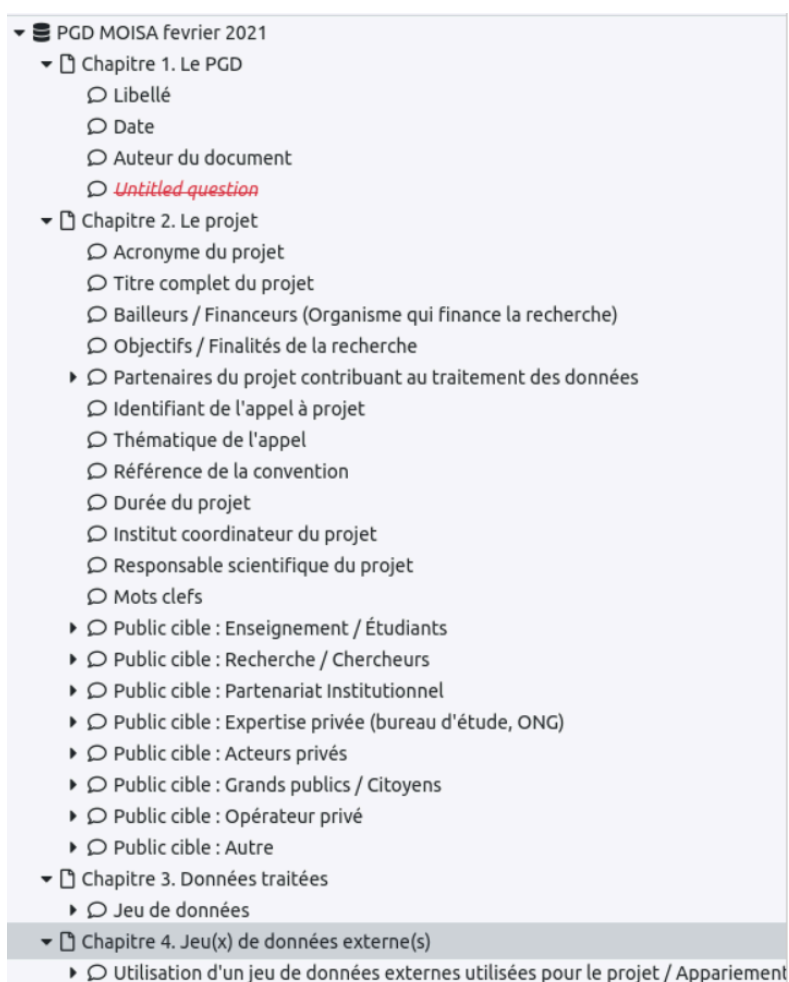
q1_1	Dummy Options Question
q1_2	Dummy Item List Question
q1_2_a_1	Sub Item List Question
q1_2_a_1_a_1	Sub Sub Options Question
q1_2_a_1_b_1	Sub Sub Options Question
q1_2_a_2	Sub Options Question
q1_2_b_1	Sub Item List Question
q1_2_b_1_a_1	Sub Sub Options Question
q1_2_b_2	Sub Options Question
q1_3	Dummy Value Question
q2_1	Other Dummy Question

Ce glossaire utilise la même base réursive pour traverser le Questionnaire.

### 5.2.4. Version définitive du format CSV

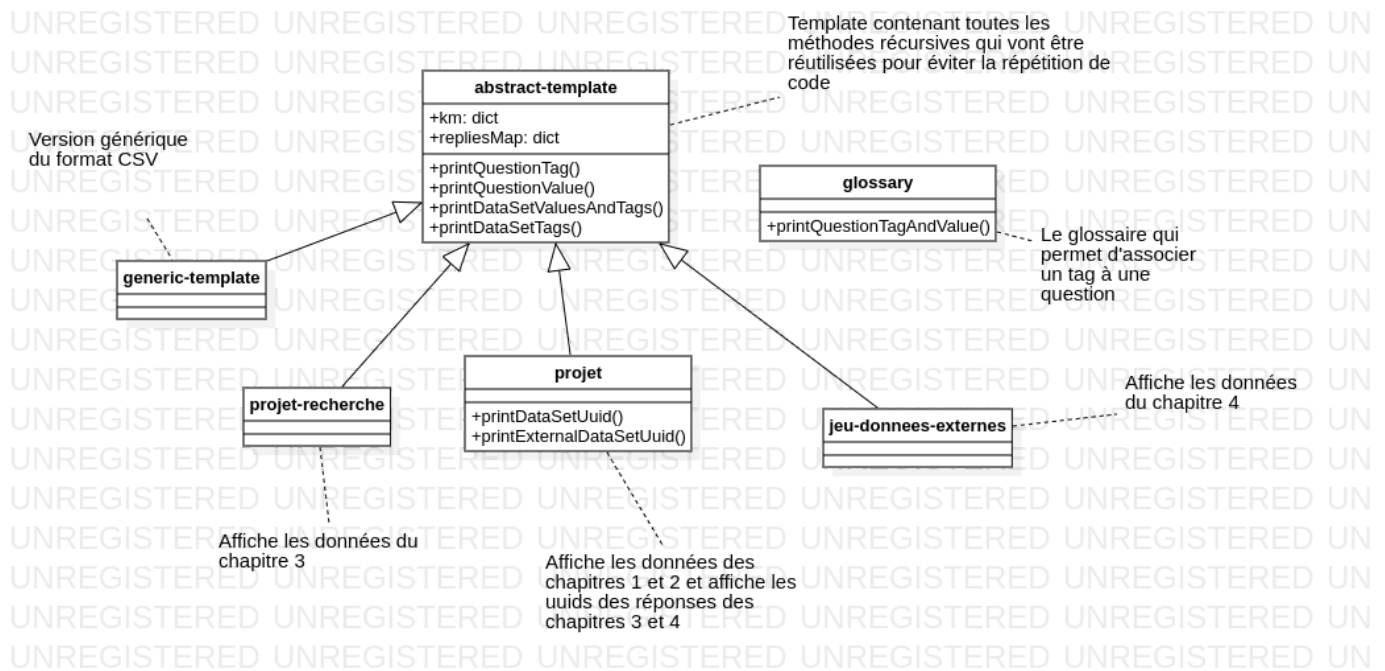
Dans le format CSV générique, on peut constater que toutes les informations sont affichées sur uniquement deux lignes : La 1<sup>re</sup> ligne correspond aux tags des questions et la deuxième aux valeurs des questions. Pour des KM, comme celui de MoISA, le format générique peut retourner un fichier CSV contenant plusieurs centaines de colonnes dû à sa grande taille. Voici un lien vers un export de format CSV générique sur un Questionnaire se basant sur le KM de MoISA : <https://docs.google.com/spreadsheets/d/1T1r8thKGm2jO1w537pLgqigfcLCVhi57SY2zC2vNbsl/edit?usp=sharing>

Voici la structure du KM de MoISA :



Mme Aubert m'a demandé de séparer les données dans plusieurs fichiers pour faciliter davantage le traitement des données du Questionnaire. Je vais vous présenter la structure que j'ai décidé d'implémenter pour séparer les données dans plusieurs fichiers grâce à ce diagramme :





- **abstract-template** contient les variables et méthodes qui vont être réutilisés par pratiquement toutes les autres templates. Jinja2 offre la possibilité d'héritage de template avec la syntaxe suivante :

```

{%- in abstract-template.csv.jinja2 -%}
{%- set km = ctx.knowledgeModel -%}
{%- set repliesMap = ctx.questionnaireReplies -%}

{%- Code des méthodes -%}
...
{%- block body -%}{%- endblock -%}
  
```

Et voici le code dans la template **projet** permettant d'hériter **abstract-template** :

```

{%- extends "src/csv/abstract-template.csv.jinja2" -%}
{%- block body -%}
{%- Code de la template projet -%}
...
{%- endblock -%}
  
```

- **projet** contient les réponses des chapitres 1 et 2. Il contient également une liste des UUIDs de chaque réponse des chapitres 3 et 4.
- **projet-recherche** contient les réponses du chapitre 3.
- **jeu-donnees-externes** contient les réponses du chapitre 4.
- **generic-template** est le [format générique](#).

Voici le lien Google Sheet vers le résultat final des différents formats CSV, je n'ai pas répondu à toutes les questions figurant dans le Questionnaire pour vous montrer à quoi cela ressemble en format CSV :

<https://docs.google.com/spreadsheets/d/1aMYNGeZsEAzcA-j4Y46cqioPNCx41eS0RSVP0bWQCoY/edit?usp=sharing>

Si vous souhaitez regarder le code du Document Template final de plus près, je vous invite à visiter ce dépôt GitHub :

<https://github.com/DSW-INRAE/document-templates>.

Celui-ci possède le format CSV que j'ai développé et également le format DOCX développé par Jocelin.

## 5.3. Service de stockage sous Nextcloud

INRAE possède une propre instance Nextcloud accessible par tous les utilisateurs d'INRAE. Voici le lien vers celle-ci : <https://nextcloud.inrae.fr>. Je vais donc travailler sur cette instance tout au long de cette deuxième mission.

### 5.3.1. Test de l'API Nextcloud

Pour communiquer avec Nextcloud à travers un service, je vais avoir besoin d'une API : l'API WebDAV<sup>[16]</sup>, me permettra de gérer les fichiers et les dossiers stockés sur Nextcloud. Je vais m'approprier cette API en utilisant encore une fois Postman. Mme Aubert m'a partagé un dossier Nextcloud sur lequel je vais pouvoir travailler et effectuer mes tests.

Voici le lien de ce dossier partagé :

<https://nextcloud.inrae.fr/s/sqmqMHzi83zPP6T>

Il nécessite également un mot de passe pour accéder à ce dossier.

Voici un exemple de requête HTTP permettant de créer un dossier dans ce dossier :

```
MKCOL /public.php/webdav/testFolder HTTP/1.1
Host: nextcloud.inrae.fr
Authorization: Basic c3Ftc...
Cookie: __Host-nc_sameSiteCookielax=true;
__Host-nc_sameSiteCookiestrict=true
```

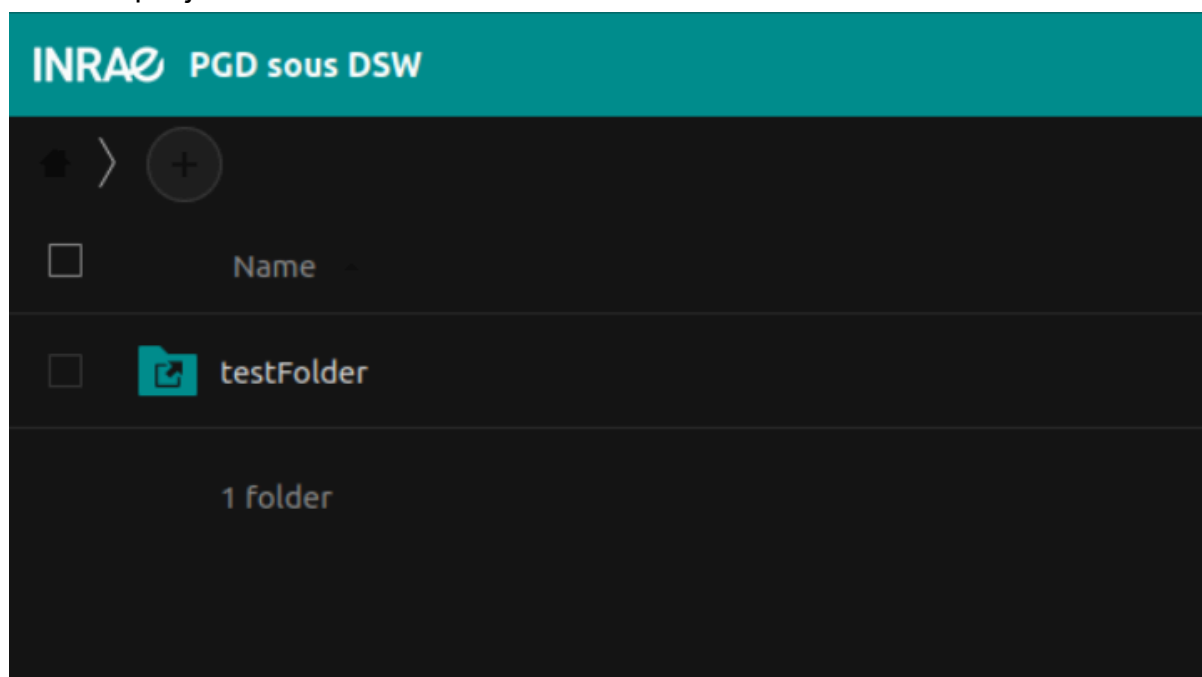
Vous pouvez constater que le point d'accès à l'API est `/public.php/webdav/`.

`testFolder` correspond au nom du dossier que je souhaite créer. Grâce au Basic Auth Header généré à partir de l'identifiant du fichier partagé (partie rouge du lien ci-dessus) et son mot de passe d'accès, Nextcloud sait que je fais référence au dossier partagé par Mme Aubert. Cette outil de génération de Basic Auth Header peut être utile pour mieux comprendre le fonctionnement d'une autorisation Basic :

<https://www.blitter.se/utills/basic-authentication-header-generator/>

Quand j'effectue cette requête, je reçois un Status Code 201 signifiant que le dossier a bien été créé.

Voici ce que je trouve sur l'interface Nextcloud :



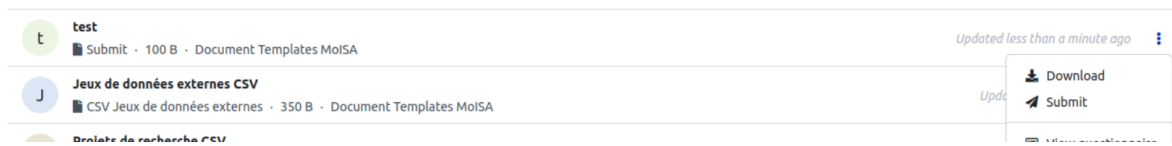
### 5.3.2. Développement du service

DSW propose une fonctionnalité très intéressante : la soumissions de document. Cette fonctionnalité permet aux utilisateurs de configurer un service permettant d'effectuer une requête HTTP, ce service est lié à un format d'export et envoie les données de ce format dans le corps de la requête. Il est donc question de développer une API pour que le service configuré sur DSW puisse communiquer avec celle-ci. Le service enverrait les informations nécessaires stockées dans le document pour que l'API puisse effectuer les traitements permettant stocker les différents formats d'export sur Nextcloud

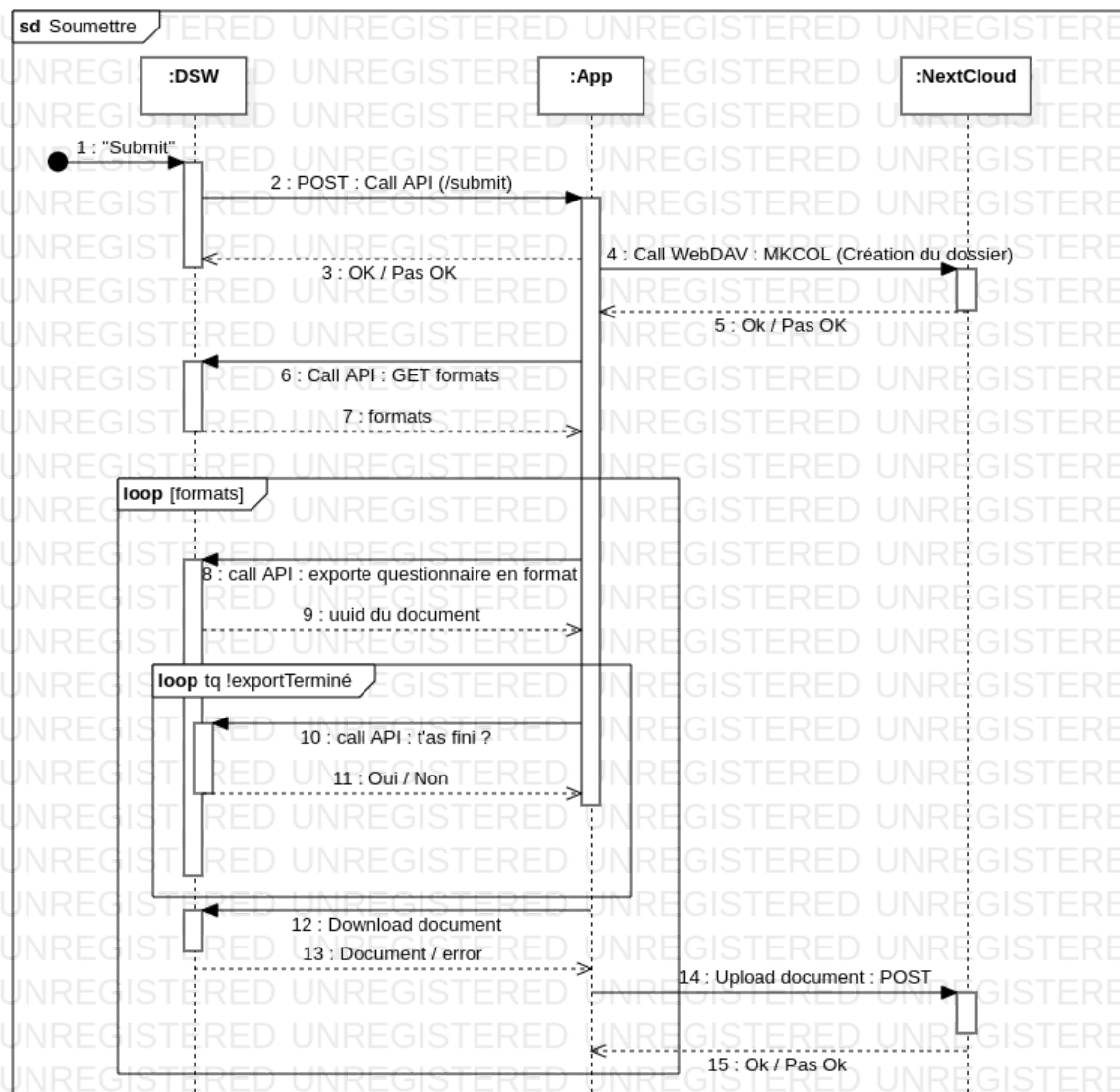
J'ai donc choisi de créer un nouveau format d'export de type json dédié à cette fonctionnalité. Ce fichier contient l'UUID et le nom du Questionnaire concerné par l'export. Exemple :

```
{
  "questionnaireUuid": "9fb99155-7a3f-4ada-9468-805d4079652a",
  "questionnaireName": "test"
}
```

Ce format peut donc utiliser la fonctionnalité **submit** de DSW :



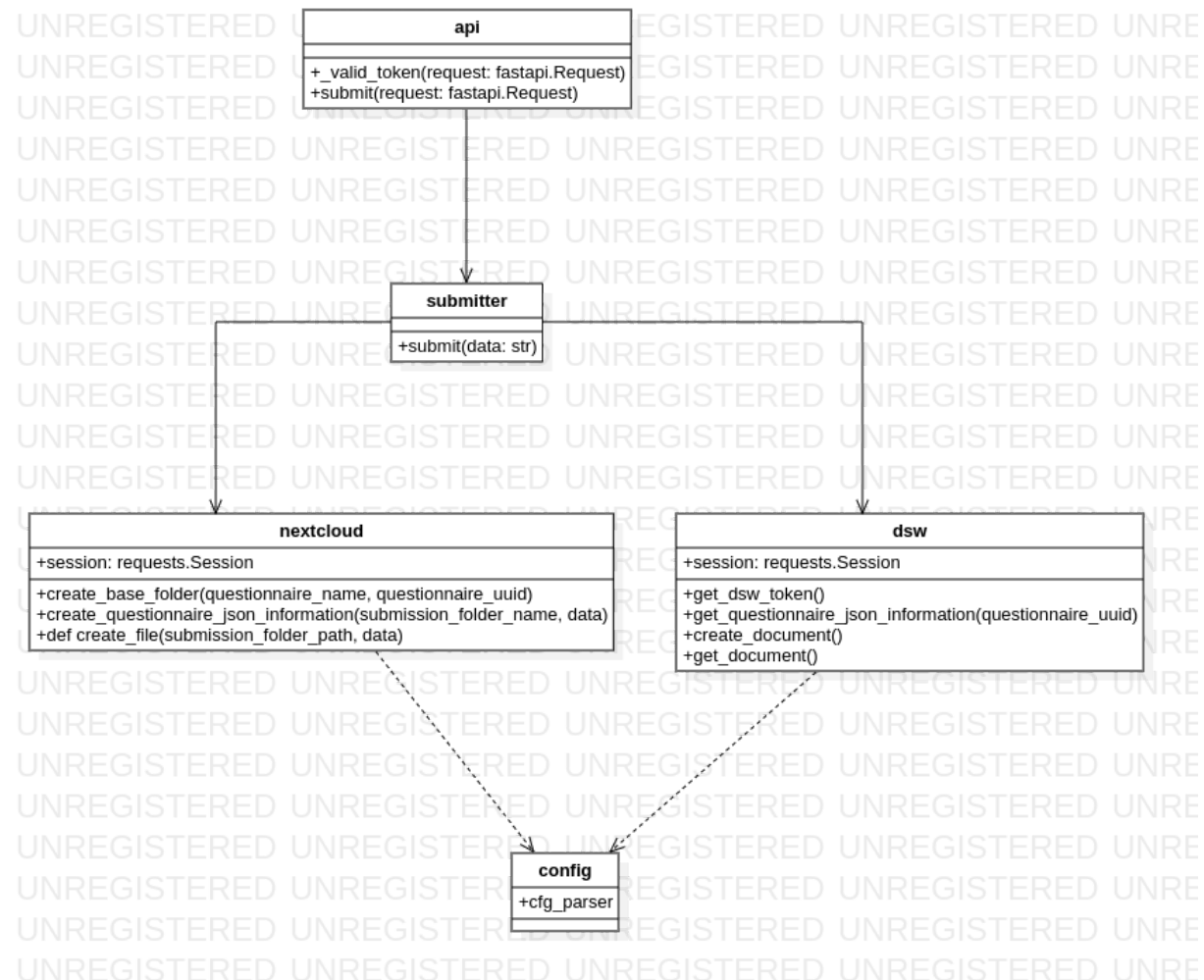
Voici un diagramme de séquence illustrant comment se déroule le processus de stockages des différents formats sur Nextcloud :



Le point de départ : le service de soumission configuré sur DSW fait un appel à l'API de stockage. Celle-ci crée un dossier sur Nextcloud grâce à son API WebDAV. Tous les fichiers liés au Questionnaire seront stockés dans ce dossier. Puis, l'API récupère tous les formats du Questionnaire, exporte chaque format en document, attend que l'exportation se finisse. Quand c'est le cas, elle télécharge le document et l'upload vers Nextcloud dans le dossier créé auparavant.

Pour le développement de cet API, j'ai choisi un outil populaire : FastAPI. Cet outil est un framework web moderne et rapide permettant de construire des applications API en utilisant Python. Il offre une performance exceptionnelle. FastAPI est conçu pour être simple à utiliser et offre une meilleure productivité aux développeurs grâce à sa syntaxe déclarative et intuitive.

Voici la structure de cette API :



La classe **submitter** va agir comme intermédiaire entre les classes **dsw** et **nextcloud**. La classe **dsw** possède toutes les méthodes liées à l'API de DSW. De même, **nextcloud** possède toutes les méthodes liées à l'API de Nextcloud. J'ai essayé de créer une structure simple, mais qui respectent tout de même les principes SOLID, dans ce cas le principe de responsabilité unique.

Voici une partie du code de la méthode **submit** de la classe **submitter** :

```
def submit(data) -> JsonResponse:
    # Useful variables
    data_dictionary = json.loads(data)
    questionnaire_uuid = data_dictionary['questionnaireUuid']
    questionnaire_name = data_dictionary['questionnaireName']
    # Creating the DMPs main folder within Nextcloud
    submission_folder_name = nextcloud.create_base_folder(questionnaire_uuid,
questionnaire_name)
    # Uploading the DSW DMPs Json file to Nextcloud
    data = dsw.get_questionnaire_json_information(questionnaire_uuid)
    nextcloud.create_questionnaire_json_information(submission_folder_name, data)
```

Le paramètre **data** correspond aux données figurants dans le document envoyé par le service configuré dans DSW. Il suffit de convertir ces données JSON en dictionnaire Python pour accéder à l'UUID et le nom du Questionnaire.

La méthode se charge de créer le dossier dans lequel se trouvera tous fichiers liés aux Questionnaires. Regardons le code de cette méthode de plus près :

```
def create_base_folder(questionnaire_name, questionnaire_uuid):
    submission_folder_name = f'{questionnaire_uuid}_{questionnaire_name}'
    url = f'{cfg.cloud_service.host}/public.php/webdav/{submission_folder_name}'
    session.request("MKCOL", url)
    return submission_folder_name
```

Cette méthode crée un variable contenant le nom du dossier qui possède la structure suivante : {uuid du Questionnaire}\_{nom du questionnaire}. Puis, elle effectue une requête HTTP en utilisant la librairie **requests** de Python. L'URL est construit à partir de la configuration de l'API et du nom du dossier.

Voici le code de la méthode permettant de récupérer le Token d'authentification qui permettra à l'API d'effectuer d'autre appel vers DSW :

```
def get_dsw_token() -> str:
    if 'dsw_token' not in globals():
        data = json.dumps({
            'code': None,
            'email': cfg.dsw.login,
            'password': cfg.dsw.password
        })
        headers = {
            'Content-Type': 'application/json'
        }
        response = requests.post(f'{dsw_base_url}/tokens', headers=headers,
                                data=data)
        global dsw_token
        dsw_token = response.json()['token']
    return dsw_token
```

J'ai voulu éviter de faire un appel à l'API à chaque fois que j'utilise la méthode `get_dsw_token()` car cela pourrait considérablement ralentir le processus de stockage. J'ai donc choisi d'implémenter un singleton.

FastAPI se charge de créer un dictionnaire à partir d'un fichier config.yml accessible à travers la classe **config**. Voici la structure de ce fichier :

```
cloud-submission-service:
  service:
    code: ok

  security:
    enabled: true
    tokens:
      - '...'

  dsw:
    login: 'albert.einstein@example.com'
    password: 'password'
    host: 'http://dsw-server'
    port: 3000

  cloud-service:
    host: 'https://nextcloud.inrae.fr'
    login: 'sqmqMHZj83zPP6T'
    password: '...'

#Logging:
#  level: WARNING
#  format: ...
```



Ce fichier de configuration permet de définir des variables telles que le lien vers l'API DSW et Nextcloud. Ceci offre la possibilité à d'autres développeurs de réutiliser le code de cette API et l'adapter à leur contexte de développement.

Si vous souhaitez regarder le code de l'API de plus près, je vous invite à visiter ce dépôt GitHub :

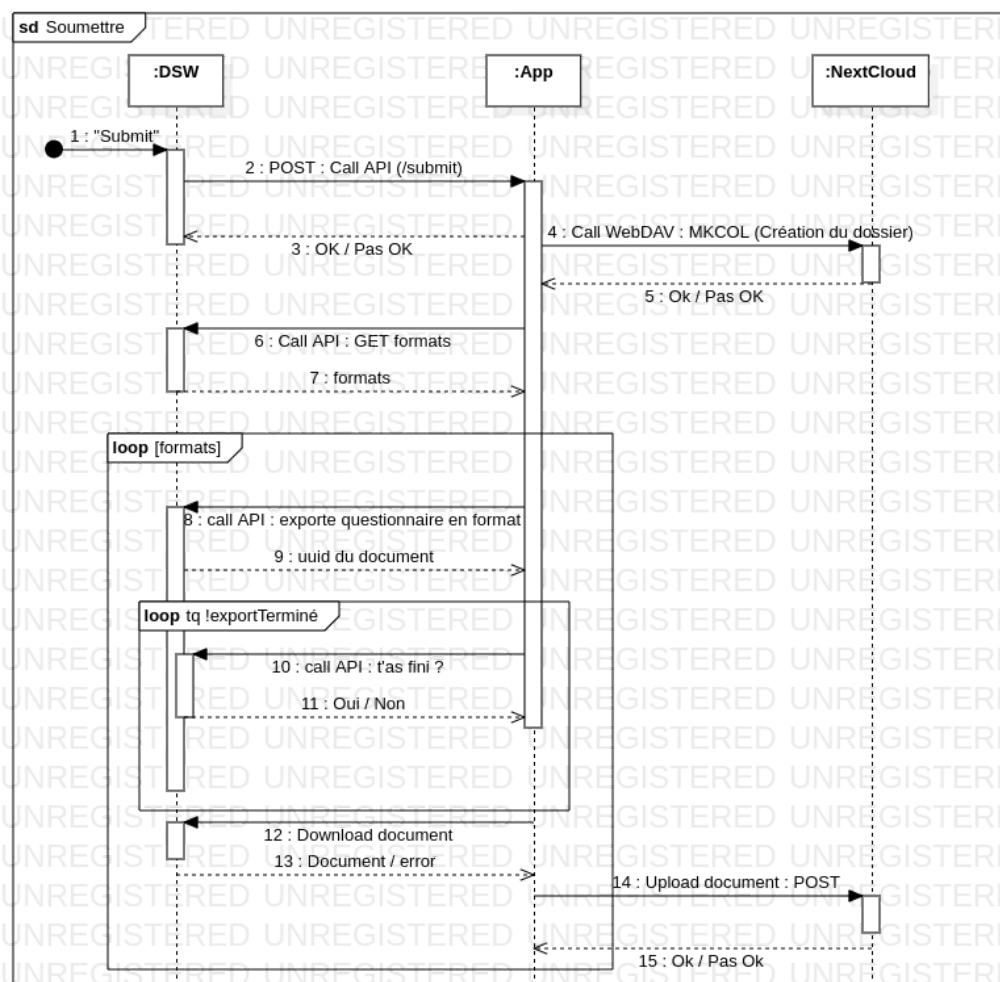
<https://github.com/DSW-INRAE/nextcloud-submission-service>.

## 6. Rapport CDD

Durant mon CDD d'un mois à INRAE, j'ai, on m'a demandé de poursuivre le travail autour des extensions développées pendant le stage. Cette partie documente les nouvelles fonctionnalités que j'ai conçues et mises en œuvre au cours de ce contrat.

### 6.1. Problème de performance concernant 'nextcloud-submission-service'

Durant mon contrat à INRAE, j'ai constaté que l'efficacité de l'API `nextcloud-submission-service` posait certain problème. En particulier, par rapport aux performances du processus d'exportation et de téléchargements des fichiers liées au PGD vers Nextcloud. Le diagramme de séquence ci-dessous offre une vue détaillée de ce processus d'exportation, je vais me baser sur celui-ci pour mettre en évidence les raisons de cette mauvaise performance.



Considérons un scénario avec un questionnaire possédant 7 formats, et faisons le décompte du nombre d'appels vers les différentes APIs lors du processus de soumission :

- 1 appel vers l'application de DSW
- 1 appel MKCOL vers Nextcloud
- 1 appel GET formats vers DSW
- 7 appels vers DSW pour exporter le format en document
- Au moins 1 appel (ou plus) pour vérifier si l'exportation a été effectuée, avec une attente de 0.2 seconde, répétée 7 fois
- 7 appels de téléchargement vers Minio
- 7 appels d'upload vers Nextcloud

Au total 31 appels vers l'API DSW.

Cette quantité d'appels peut résulter à un temps de calcul significatif (30 ou plus secondes en local, potentiellement plus sur un serveur). Et peut même mener à un timeout si le processus ne se termine pas assez rapidement.

Afin d'améliorer l'efficacité de ce processus, j'ai envisagé plusieurs améliorations :

Premièrement, une solution propice serait de séparer et catégoriser les formats. Cette approche implique donc l'exécution de soumissions de documents distinctes pour chaque type de format (CSV, Docx, JSON). Cette stratégie a pour bénéfice d'éliminer tout problème potentiel de timeout en considérant que cela réduit considérablement le nombre d'appels vers DSW. Elle répond par ailleurs aux défis liés à la gestion des fichiers associés aux données personnelles génétiques (PGD), simplifiant ainsi le processus. Concrètement, trois dossiers distincts sur Nextcloud ont été dédiés à chaque type de format, facilitant ainsi le stockage et l'organisation des données.

Une autre possibilité d'amélioration serait de déléguer la création des documents CSV à l'API `nextcloud-submission-service`, en utilisant le format JSON fourni par l'API de DSW. Cela minimisera, non seulement, le nombre d'appels à l'API de DSW (plus besoin d'exporter ni de télécharger les documents à partir du MinIO), mais contribuera également à un runtime plus court dû au fait que DSW utilise Jinja2 pour la création de fichier qui n'est pas le langage le plus efficace pour effectuer ce type de tâches. Cette stratégie vise à optimiser l'ensemble du processus et à améliorer la lisibilité et la compréhension du code, en remplaçant l'utilisation parfois lente et complexe de Jinja2.

Dans le cadre de cette optimisation, j'ai choisi de créer une nouvelle API qui englobera les fonctionnalités de l'API `nextcloud-submission-service`. Cependant, cette API a été conçue pour être extensible, permettant l'ajout d'autres

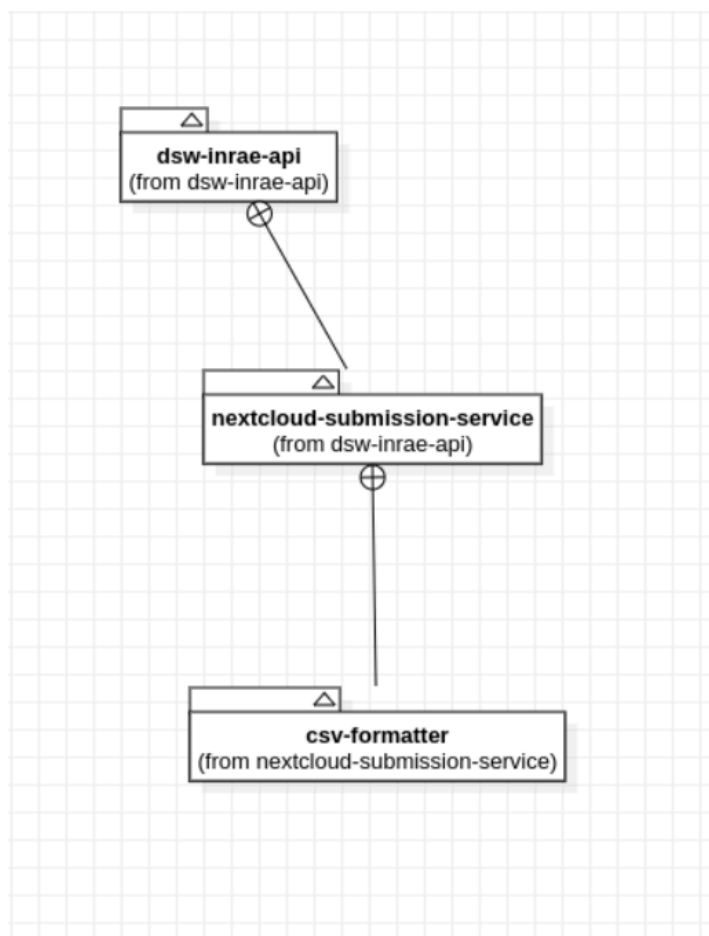
extensions potentielles, que je détaillerai ultérieurement. Cette approche garantit la flexibilité de l'outil tout en offrant une base solide pour les développements futurs.

## 6.2. Développement de 'dsw-inrae-api'

Le développement de l'API `dsw-inrae-api` vise donc à optimiser l'ensemble du processus de soumission de document. Cette API, développée avec FastAPI, un framework web réactif basé sur Python, englobe l'ancienne API `nextcloud-submission-service` sous forme de module. J'ai également été amené à effectuer une refonte du module `nextcloud-submission-service`, qui, à présent, possède la logique de création des fichiers CSV directement en code Python et n'utilise donc plus DSW pour la création des fichiers CSV.

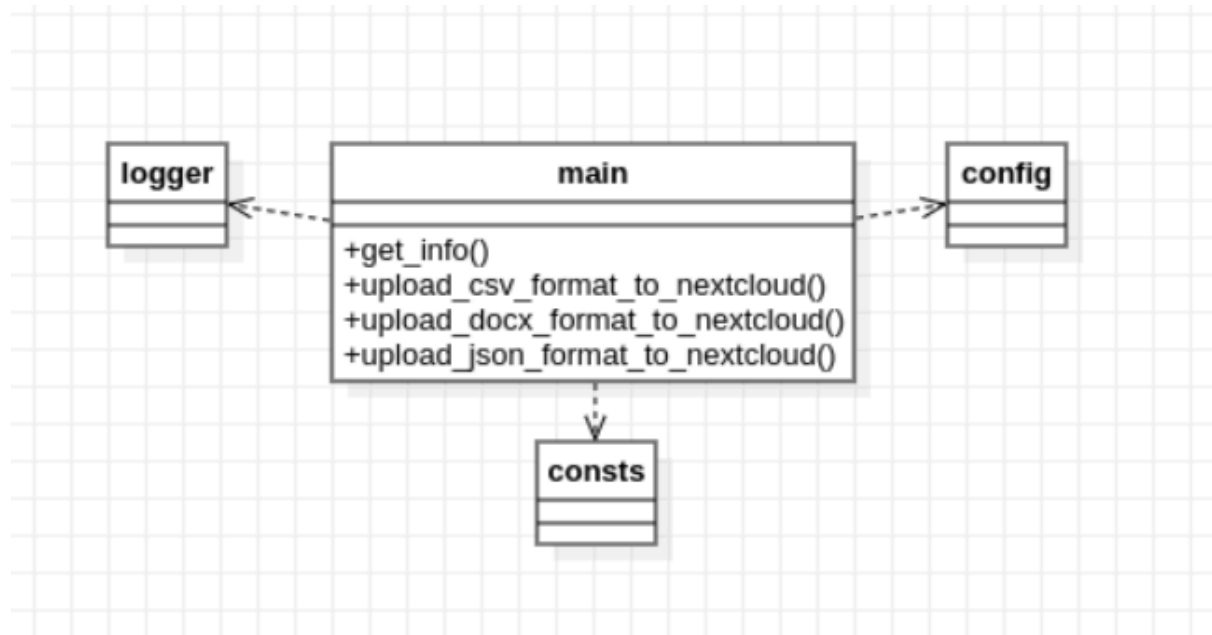
Dans les sections suivantes, je vais présenter des extraits de code et des diagrammes pour offrir une vision détaillée de la structure de l'API `dsw-inrae-api`.

Voici la structure générale de cette API :



On peut constater que cette nouvelle API englobe donc l'ancienne API `nextcloud- submission-service`, et que le module `csv-formatter` qui se charge de créer les fichiers CSV est intégré en tant que module contenu dans `nextcloud-submission-service`.

Voici un aperçu de la structure du module principal de `dsw-inrae-api` :



Les classes `logger`, `config`, `consts` servent à configurer divers paramètres de l'API, assurant une gestion centralisée des configurations. Elles sont utilisées par la classe `Main`, qui agit comme le point d'entrée de l'API.

```
Theodore
@ app.post("/nextcloud/csv")
> async def upload_csv_format_to_nextcloud(request: fastapi.Request):...

Theodore
@ app.post("/nextcloud/docx")
> async def upload_docx_format_to_nextcloud(request: fastapi.Request):...

Theodore
@ app.post("/nextcloud/json")
> async def upload_json_format_to_nextcloud(request: fastapi.Request):...
```

Chaque méthode de la classe `Main` est dédiée au téléchargement d'une catégorie spécifique de fichier vers Nextcloud.

Du côté de DSW, il suffit de s'assurer qu'il existe trois objets "Document Submission", chacun étant dédié à la soumission d'un format spécifique. Ces objets font respectivement appel aux routes correspondantes à chaque format défini dans la classe `Main`.

Rentrons plus en détail dans le code spécifique aux méthodes de soumission pour les différents formats dans la classe `Main` :

```
@ app.post("/nextcloud/csv")
async def upload_csv_format_to_nextcloud(request: fastapi.Request):
    ...
    data = await request.body()
    submit_strategy = CsvSubmitStrategy()
    submitter = Submitter(submit_strategy)
    return submitter.submit(data)

@ app.post("/nextcloud/docx")
async def upload_docx_format_to_nextcloud(request: fastapi.Request):
    ...
    data = await request.body()
    submit_strategy = DocxSubmitStrategy()
    submitter = Submitter(submit_strategy)
    return submitter.submit(data)

@ app.post("/nextcloud/json")
async def upload_json_format_to_nextcloud(request: fastapi.Request):
    ...
    # (2) Get data and code
    data = await request.body()
    submit_strategy = JsonSubmitStrategy()
    submitter = Submitter(submit_strategy)
    return submitter.submit(data)
```

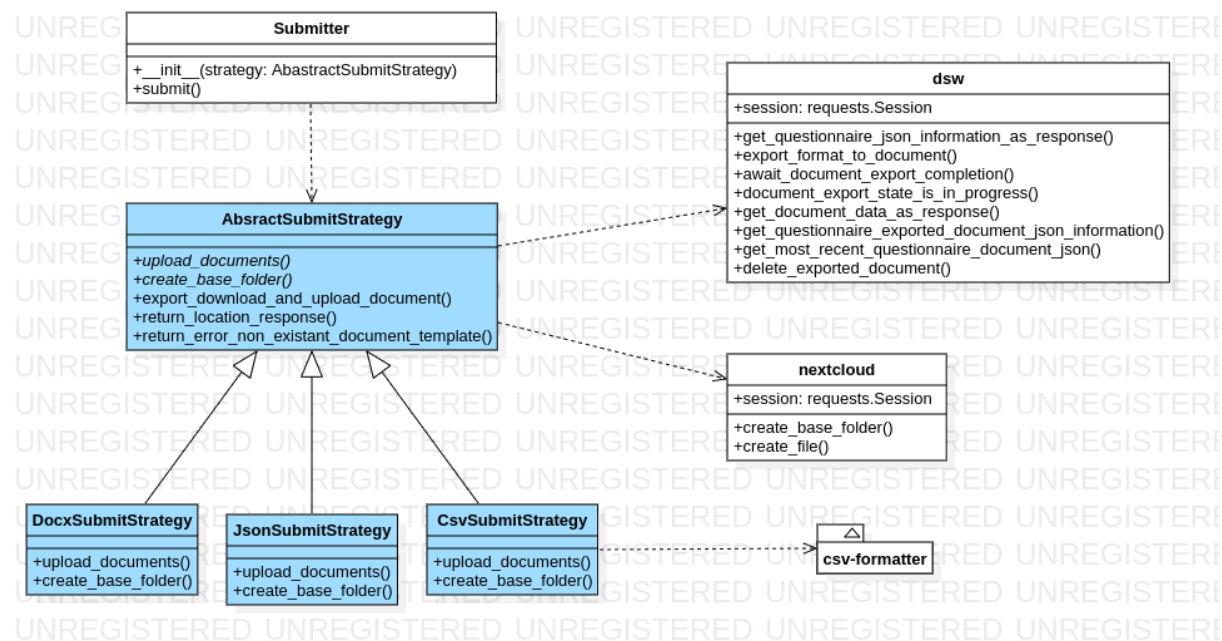
On observe que la seule ligne de code qui diffère entre ces méthodes est l'initialisation de la variable `submit_strategy`. Pour la méthode gérant le format CSV, un nouvel objet de type `CsvSubmitStrategy` est créé, tandis que pour la méthode gérant le format DOCX, un nouvel objet de type `DocxSubmitStrategy` est instancié, de même pour le format JSON. Cette approche reflète en réalité l'application d'un patron de conception connu sous le nom de "Strategy Pattern".

Ce patron permet de déléguer la logique de soumission à la classe de stratégie appropriée, rendant ainsi le choix du format interchangeable. L'utilisation de ce patron de conception offre une flexibilité significative, permettant d'ajouter facilement

de nouveaux formats sans modifier le code existant, et en favorisant une structure modulaire et facilement testable.

### 6.3. Nouvelle structure de 'nextcloud-submission-service'

En effet, c'est le module `nextcloud-submission-service` qui intègre le "Strategy Pattern" pour gérer les différents formats d'exportation. Je vais donc vous illustrer la nouvelle structure de ce module à l'aide du diagramme suivant :

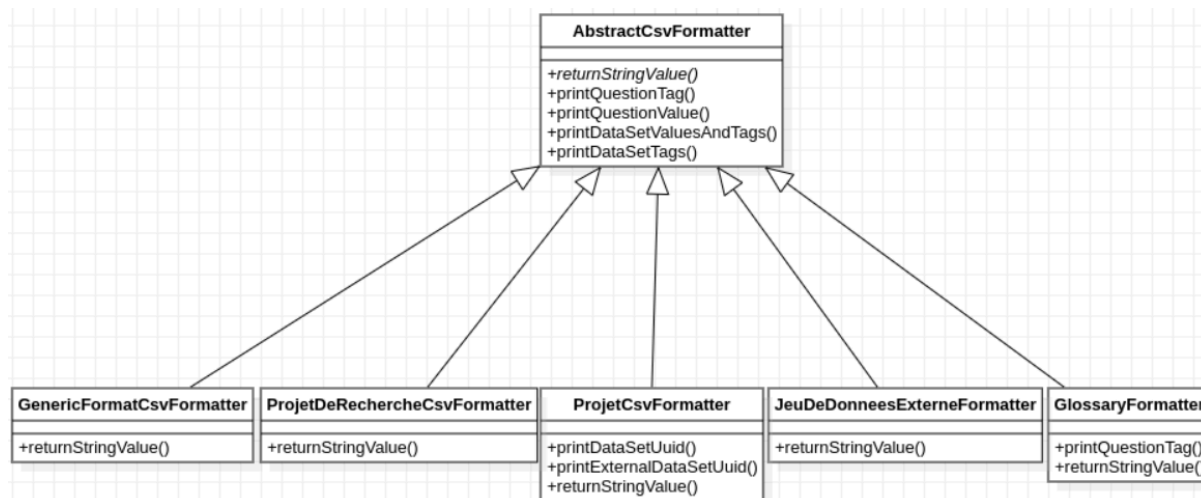


La section bleue du diagramme représente le "Strategy Pattern". Dans ce schéma, les méthodes `upload_document` et `create_base_folder` de la classe `AbstractSubmitStrategy` seront à redéfinir dans les classes concrètes telles que `DocxSubmitStrategy`, etc. Les autres méthodes de `AbstractSubmitStrategy` sont des méthodes par défaut qui seront utilisées par toutes les classes concrètes. Elles se chargent notamment d'appeler les méthodes contenues dans les classes `dsw` et `nextcloud`, qui elles effectuent respectivement se chargent de faire les appels aux API de DSW et Nextcloud.

La classe `Submitter` est le point d'entrée du module `nextcloud-submission-service` et prend une stratégie en paramètre lors de sa construction. Cette stratégie est fournie dans la classe `Main` présentée auparavant.

## 6.4. Le module 'csv-formatter'

La classe `CsvSubmitStrategy`, responsable de la création des fichiers CSV et de leur téléchargement vers Nextcloud, utilise le module `csv-formatter` pour générer ces fichiers. Je vais donc vous présenter la structure de ce module.



La classe `AbstractCsvFormatter` possède une méthode abstraite, `returnStringValue`, qui doit être redéfinie par les classes concrètes. Chaque classe concrète, comme `ProjetCsvFormatter`, correspond à un fichier CSV qui sera stocké sur Nextcloud. Le code de ce module est similaire à celui utilisé dans le code Jinja2 des modèles de documents templates développés lors du stage précédent :

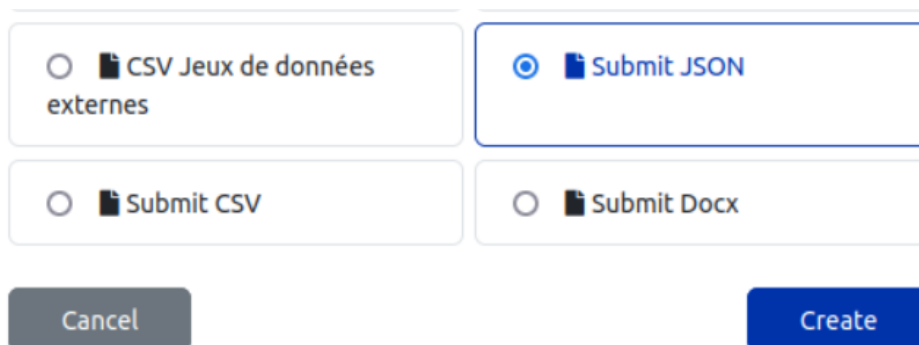
<https://github.com/DSW-INRAE/document-templates>.

Cependant, comme mentionné précédemment, le fait de créer les documents CSV du côté de l'API, en utilisant le format JSON fourni par l'API de DSW, permet non seulement de minimiser le nombre d'appels à l'API de DSW, mais contribue également à un temps d'exécution plus rapide.

En ce qui concerne le code de création des fichiers CSV dans l'API, il serait pertinent d'explorer des design patterns potentiels pour renforcer l'extensibilité et la flexibilité du code. Deux patterns qui pourront particulièrement contribuer à cette amélioration sont le "**Composite Pattern**" et "**l'Iterator Pattern**". Ces patterns offriront une structure modulaire et évolutive, rendant le code plus ouvert à d'éventuelles additions de formats de questions de la part de DSW.



Du côté de DSW, j'ai mis en place trois formats distincts qui font appel à leurs "Document Submission" respectifs. Il vous suffit de vous rendre dans la section "documents" d'un projet et de créer un nouveau document de type "submit".



☐ CSV Jeux de données externes

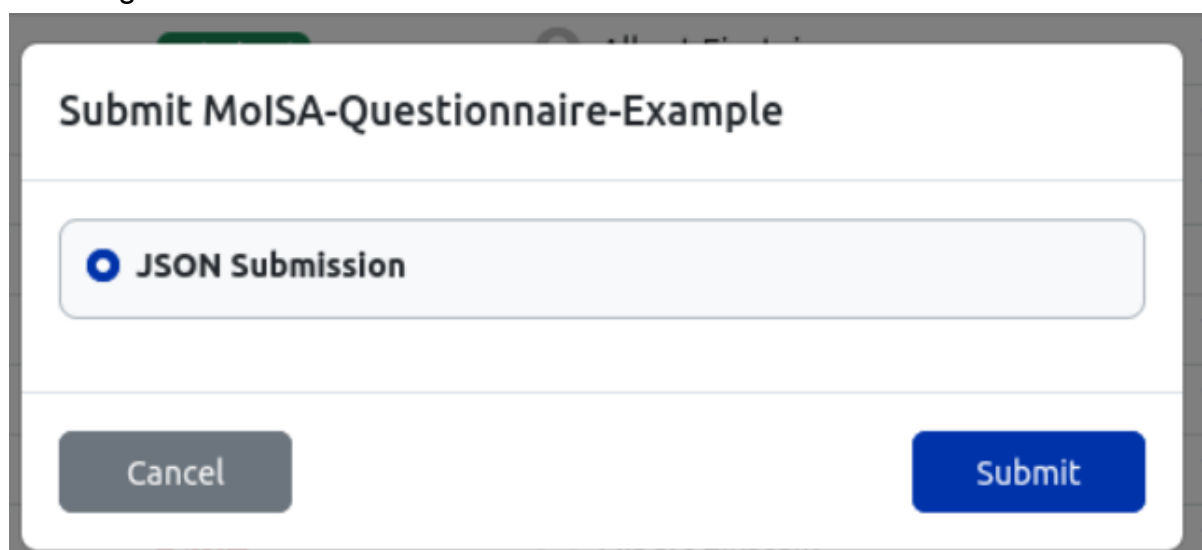
☒ Submit JSON

☐ Submit CSV

☐ Submit Docx

Cancel Create

Sur ce document, vous pouvez utiliser la fonction "submit" qui fera appel à l'API et téléchargera les documents ciblés vers Nextcloud.

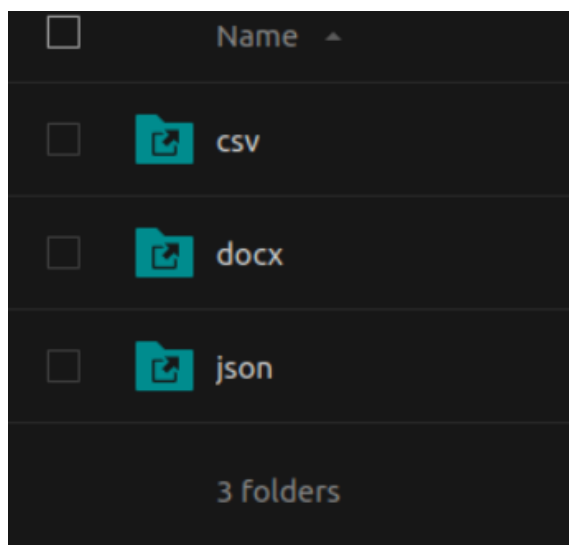


Submit MoISA-Questionnaire-Example

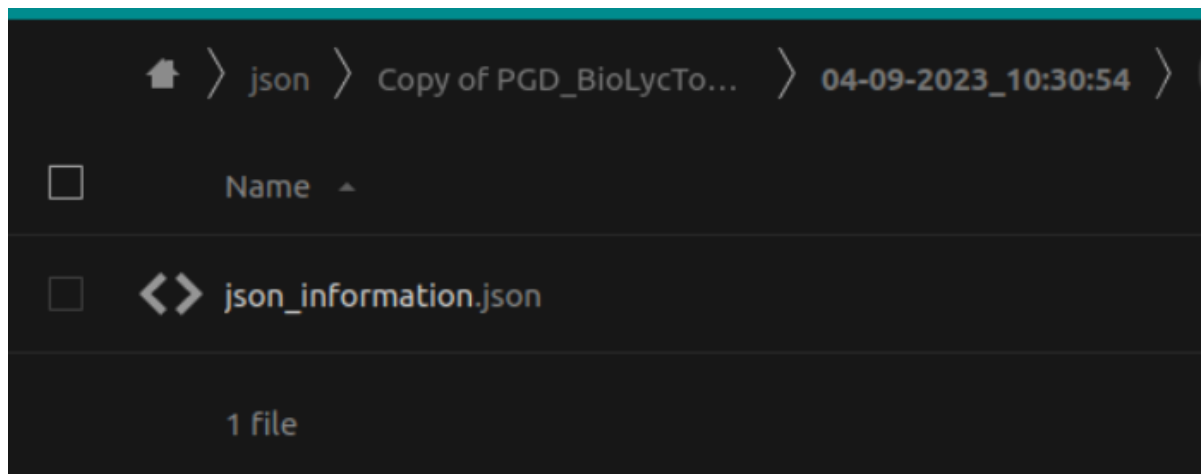
☒ JSON Submission

Cancel Submit

Les dossiers correspondant à chaque catégorie de fichiers sont organisés sur Nextcloud de la manière suivante :



Et voici le fichier JSON qui a été téléchargé sur Nextcloud :



## 6.5. Extensions Envisageables

Cette API pourrait englober d'autres extensions, comme une base de données contenant des informations sur le Plans de Gestion des Données. Cette base de données, orientée vers la Science Ouverte, permettrait à d'autres chercheurs ne provenant pas d'INRAE d'accéder et de consulter les détails des projets de recherches effectuées par INRAE, offrant une transparence dans les activités de recherche. Et éventuellement offrir la possibilité aux chercheurs externes de contacter directement les agents d'INRAE concernés pour de possibles collaborations.

Il serait également envisageable de mettre en place une interface conçue pour explorer et consulter les données exposées des PGD, en utilisant, bien-sûr, l'API [dsw-inrae-api](#). Cette interface permettrait aux chercheurs externes d'interagir visuellement avec les données, facilitant l'exploration de la base de données et améliorant l'expérience utilisateur.

En ajoutant ces extensions, l'API élargirait son champ d'application, passant de la simple soumission de documents à une plateforme plus complète, favorisant la collaboration, la transparence et l'accessibilité des informations au sein de la communauté de recherche.

## 7. Méthodes de travail et gestion du projet

Pour la gestion et l'organisation de notre stage, Jocelin et moi avons utilisé plusieurs outils qui nous ont été très utiles pour la communication et la planification de nos tâches. Ces outils nous sont déjà familiers, car nous les avons utilisés au préalable lors de projets universitaires. Ceci nous a donc permis de les adopter facilement et efficacement pour notre stage.

Durant notre stage, nous avons eu des réunions régulières avec nos tuteurs, Mme Aubert et M. Rousselle. Ces échanges constructifs nous ont permis d'obtenir un feedback continu sur nos progrès et d'ajuster nos tâches en conséquence.

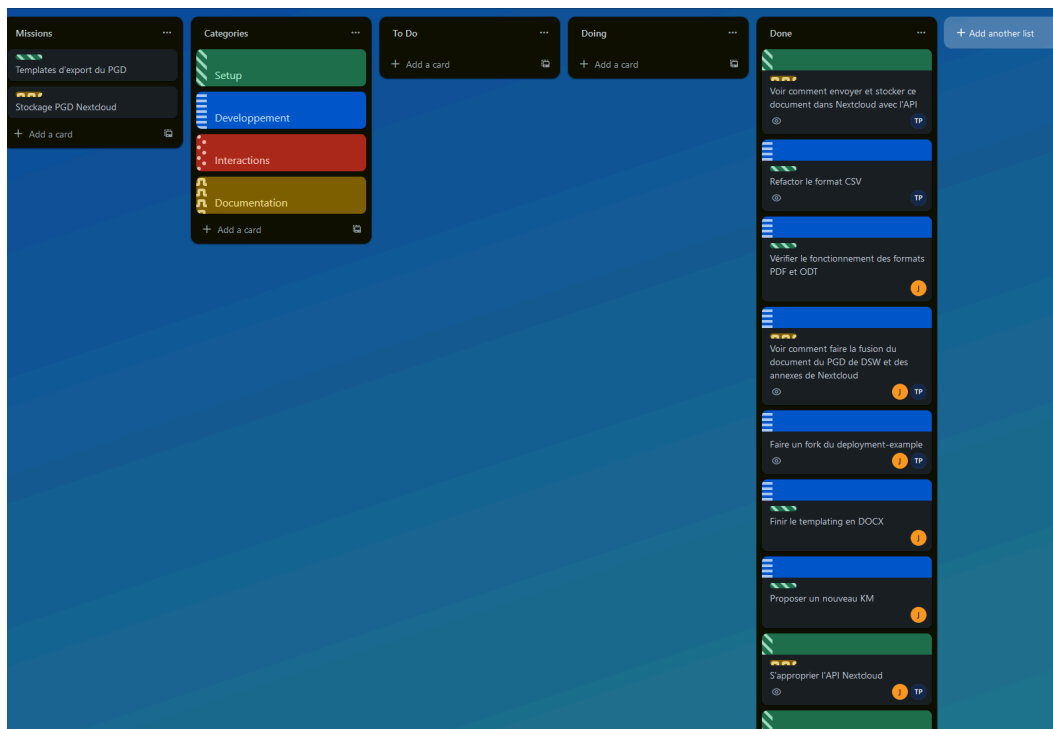
Jocelin et moi avons travaillé de manière plutôt autonome. Étant donné que nous étions seulement deux à travailler sur ce projet, nous avons choisi de mettre en place une organisation relativement simple, adaptée à nos besoins spécifiques.

### 7.1. Trello et Scrum

Pour mener à bien notre stage, nous avons choisi de suivre une version simplifiée de la méthode agile SCRUM\*.

SCRUM est une méthode agile de gestion de projet, principalement utilisée dans le développement logiciel. Elle se base sur une approche itérative, incrémentale et collaborative, visant à maximiser la productivité et la qualité du travail réalisé. Cette méthode se repose sur un concept de cycles de développement courts appelés "sprints", généralement d'une durée de 1 à 4 semaines. SCRUM met l'accent sur la transparence, la collaboration et l'adaptabilité. Il favorise également une communication fréquente entre les membres de l'équipe et encourage l'amélioration continue du processus de développement.

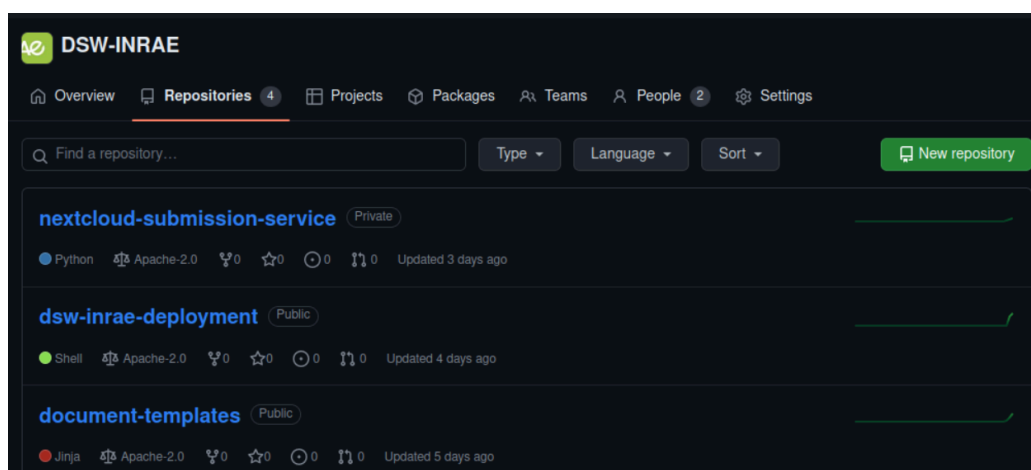
Un outil très puissant pour le SCRUM est Trello :



Trello est un outil puissant pour le stockage et la gestion de l'état des tâches. Il permet d'attribuer des tâches à des membres d'une équipe, ce qui facilite la répartition des responsabilités. Grâce à cet outil, j'ai pu suivre facilement les progrès de Jocelin et m'ajuster à son progrès pour optimiser notre collaboration.

## 7.2. Création de notre organisation GitHub

Comme nous avons eu plusieurs missions à réaliser, nous avons choisi de créer une organisation dédiée au stage et ensuite créer des dépôts dédiés à chaque mission du stage. Voici le lien vers notre organisation : [lien](#)



- Le premier dépôt sur cette capture d'écran est dédié au développement du service de stockage sous Nextcloud.
- Le deuxième dépôt contient une version personnalisée du dépôt [dsw-deployment-example](#). Dépôt que j'ai utilisé au début de la partie résultat pour héberger localement DSW, mais nous avons choisi de forker celui-ci pour nous donner une plus grande liberté de configuration.
- Le dernier dépôt est dédié au développement des différents formats d'exportation.

# Conclusion

Dans le cadre de mon stage, Jocelin et moi avons été amenés à développer des extensions pour un outil de rédaction de PGD : DSW. Cet outil est actuellement en phase de test au sein de l'unité MoISA qui fait partie d'INRAE. Un PGD est un document évolutif qui détaille la manière dont les données sont gérées. Ce document est un élément clé permettant aux chercheurs de prouver que leurs pratiques sont conformes aux exigences réglementaires du RGPD. DSW propose des fonctionnalités et des outils puissants aux développeurs tels que leur API ou encore DSW-TDK, un outil permettant de développer ses propres formats de document. Nous avons essayé d'exploiter ces fonctionnalités au maximum de leur potentiel, ce qui nous a facilité la réalisation de nos missions.

L'objectif de la première mission était de développer des formats d'exportation du PGD grâce à DSW-TDK. Je me suis concentré sur le développement du format de feuilles de calcul (CSV), tandis que Jocelin s'est focalisé sur le développement du format d'un document formel (DOCX / PDF). Les analyses statistiques effectuées grâce au format CSV seront, pour l'unité MoISA, un argument supplémentaire qui prouvera la conformité au RGPD.

Lors du développement du format CSV, j'ai dû mettre en place un système récursif permettant de pallier la structure arborescente et imprédictible des [Knowledges Models](#). Finalement, le format CSV a été découpé en plusieurs fichiers pour faciliter le traitement des données effectué lors des analyses statistiques.

Concernant la deuxième mission, il était question d'automatiser le stockage des documents liés au Questionnaire (PGD). Ceci a permis de centraliser l'espace de stockage des PGD saisis au niveau de l'unité. Je me suis chargé de développer l'API / le service en utilisant FastAPI, une librairie Python. Cette mission a pour but de faciliter le stockage des PGD officiel sur un même endroit

L'objectif final de ce stage est de promouvoir DSW au sein d'INRAE et d'élargir son utilisation. Ayant communiqué avec plusieurs chercheurs provenant de différentes unités et organisations, nous avons déjà donné une chance à DSW de se développer au sein de cet institut.

D'un point de vue technique, j'ai été régulièrement confronté à gérer des requêtes HTTP, des API et des données JSON. De plus, j'ai pu développer mes compétences en Python et son langage gabarit Jinja2. J'ai aussi eu la chance de construire ma propre API REST grâce à la librairie Python : FastAPI. Ceci a renforcé mes notions en développement web, que j'avais abordées lors du 3<sup>e</sup> et 4<sup>e</sup> semestre avec les UEs 3.01, 4.01, et 4.A.10. J'ai également revu la récursivité avec la mission d'export en format CSV. Concept abordé lors du 3<sup>e</sup> semestre.

Le développement des extensions a fortement orbité au tour de Docker, outil que j'ai beaucoup manipulé. Ceci m'a permis de développer davantage mes compétences Docker, compétences que j'avais acquises durant le 4<sup>e</sup> semestre en Virtualisation. Comme déjà évoqué, je me suis concentré sur la partie développement web (Requête HTTP, API Rest, manipulation de JSON) et mon collègue de stage s'est concentré sur l'aspect déploiement d'application.

Ce stage m'a également apporté beaucoup dans l'aspect du travail d'équipe et de la collaboration. J'ai dû régulièrement sortir de ma zone de confort pour mener au bon développement des missions du stage.

Finalement, nous avons essayé de nous répartir les tâches de manière équitable, pour qu'elles soient cohérentes avec nos spécialités du Bachelor Universitaire de Technologie informatique.



# Bibliographie

- [1]  
“Data Stewardship Wizard 3.22 documentation.”  
<https://guide.ds-wizard.org/en/latest/index.html>.
- [2]  
“Data Stewardship Wizard Engine Tools.” Data Stewardship Wizard, Jan. 31, 2023. Available: <https://github.com/ds-wizard/engine-tools>
- [3]  
“Datapartage - Outils et guides.”  
<https://datapartage.inrae.fr/Gener/Rediger-un-PGD/Comment-rediger-un-plan-d-e-gestion/Outils-et-guides>.
- [4]  
“Datapartage - Pourquoi rédiger un plan de gestion ?”  
<https://datapartage.inrae.fr/Gener/Rediger-un-PGD/Pourquoi-rediger-un-plan-d-e-gestion>.
- [5]  
“DipSO - Accueil.” <https://www6.inrae.fr/dipso>.
- [6]  
“Document Template Development - Data Stewardship Wizard 3.23 documentation.”  
<https://guide.ds-wizard.org/en/latest/more/development/document-templates.html>.
- [7]  
“ds-wizard/dsw-deployment-example.”  
<https://github.com/ds-wizard/dsw-deployment-example>.
- [8]  
“DSW Deployment Example.” Data Stewardship Wizard, Apr. 28, 2023  
Available: <https://github.com/ds-wizard/dsw-deployment-example>
- [9]  
“FastAPI.” <https://fastapi.tiangolo.com/>.
- [10]  
“MoISA, Montpellier Interdisciplinary center on Sustainable Agri-food systems (social and nutritional sciences).” <https://umr-moisa.cirad.fr/>.
- [11]  
“Qu’est-ce que Docker ?”  
<https://www.oracle.com/fr/cloud/cloud-native/container-registry/what-is-docker/>.

- [13]  
“Swagger UI.” <https://api-researchers.ds-wizard.org/swagger-ui/#/>.
- [14]  
“Template Designer Documentation — Jinja Documentation (3.1.x).”  
<https://jinja.palletsprojects.com/en/3.1.x/templates/#>.
- [15]  
“The DSW Story,” *Data Stewardship Wizard*. <https://ds-wizard.org/dsw-story>.
- [16]  
“Webdav — Nextcloud latest Developer Manual latest documentation.”  
[https://docs.nextcloud.com/server/22/developer\\_manual/client\\_apis/WebDAV/index.html](https://docs.nextcloud.com/server/22/developer_manual/client_apis/WebDAV/index.html) (accessed Jun. 14, 2023).

# Annexes

## docker-compose.yml de dsw-deployment-example

```
version: '3'
services:

  dsw-server:
    image: datastewardshipwizard/wizard-server:3.23.0
    restart: always
    ports:
      # (!!)Expose only for local deployment, externally use HTTPS proxy
      - 127.0.0.1:3000:3000
    depends_on:
      - postgres
      - minio
    volumes:
      - ./config/application.yml:/app/config/application.yml:ro
    extra_hosts:
      - host.docker.internal:host-gateway

  dsw-client:
    image: datastewardshipwizard/wizard-client:3.23.0
    restart: always
    ports:
      # (!!)Expose only for local deployment, externally use HTTPS proxy
      - 127.0.0.1:8080:8080
    environment:
      API_URL: http://localhost:3000

  docworker:
    image: datastewardshipwizard/document-worker:3.23.0
    restart: always
    depends_on:
      - postgres
      - minio
      - dsw-server
    volumes:
      - ./config/application.yml:/app/config/application.yml:ro
    extra_hosts:
      - host.docker.internal:host-gateway

  mailer:
    image: datastewardshipwizard/mailer:3.23.0
    restart: always
    depends_on:
      - postgres
      - dsw-server
    volumes:
      - ./config/application.yml:/app/config/application.yml:ro
```

```

postgres:
  image: postgres:15.2
  restart: always
  # (!!) Expose only for debugging locally
  # ports:
  #   - 127.0.0.1:5432:5432
  # (!!) Change default password
  environment:
    POSTGRES_DB: engine-wizard
    POSTGRES_USER: postgres
    POSTGRES_PASSWORD: postgres
  # (!!) Mount for persistent data
  # volumes:
  #   - db-data:/var/lib/postgresql/data
  # OR
  #   - ./db-data/data:/var/lib/postgresql/data

minio:
  image: minio/minio:RELEASE.2022-08-02T23-59-16Z
  restart: always
  command: server /data --console-address ":9001"
  ports:
    - 9000:9000
    - 9001:9001
  environment:
    MINIO_ROOT_USER: minio
    MINIO_ROOT_PASSWORD: minioPassword
  # (!!) Mount and backup for persistent data
  # volumes:
  #   - s3-data:/data
  # OR
  #   - ./s3-data/data:/data

# volumes:
#   db-data:
#   s3-data:

```