**Projet CAO Intégrée**

Spécifications Fonctionnelles Détaillées

Double Codification – IHM

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Nom** | **Titre** | **Date** |
| *Rédigé par :* | V. Level | CGI | 19/12/2024 |
| *Approuvé par :* | S. Blanchard | MOS |  |

Liste de Diffusion

|  |  |  |
| --- | --- | --- |
| **Nom des intervenants** | **Fonctions et Responsabilités** | **Initiale** |
| SEBASTIEN BLANCHARD | Pilote métier |  |
| DAVID VINET | Expert SI CAO |  |
| BERTRAND BOUTIN | Expert SI CAO |  |
| ARNAUD HERVY | Expert SI CAO |  |

Historique des évolutions

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Commentaires** |
| V1.0 | 19/12/2024 | Création du document |
| V1.1 | 22/01/2025 | Relecture V1.0 |

**Objectifs généraux**

La spécification détaillée, a pour objectif l’identification et le détail pour une fonctionnalité donnée :

* Du concept d’utilisation.
* Des paramétrages à effectuer.
* Des réalisations techniques à mettre en place

|  |  |  |  |
| --- | --- | --- | --- |
| **Points en suspens** | **Hyperlien dans la spécification** | **Qui doit traiter** | **Pour quand** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table des Matières**

[1. Description générale 4](#_Toc189646278)

[2. Description détaillée 5](#_Toc189646279)

[2.1. Description fonctionnelle des Cas d’utilisation 5](#_Toc189646280)

[2.1.1. Description 5](#_Toc189646281)

[2.1.2. Schéma de la base de données 6](#_Toc189646282)

[2.1.2.1. Description du schéma 6](#_Toc189646283)

[2.1.2.1.1. Tables Catalogues 6](#_Toc189646284)

[2.1.2.1.2. Table métier 6](#_Toc189646285)

[2.1.3. API Double Codification 9](#_Toc189646286)

[2.1.3.1. Généralités et paramétrages 9](#_Toc189646287)

[2.1.3.2. Descriptions des contrôleurs 9](#_Toc189646288)

[2.1.4. Job de Consolidation des données 11](#_Toc189646289)

[2.1.4.1. Généralités et paramétrages 11](#_Toc189646290)

[2.1.5. Application Double Codification 16](#_Toc189646291)

[2.1.5.1. Généralités et paramétrages 16](#_Toc189646292)

[2.1.5.2. Fonctionnalitées principales 16](#_Toc189646293)

[2.1.5.3. Accès à l’application 16](#_Toc189646294)

[2.1.5.4. Ecran principal de l’application 17](#_Toc189646295)

[2.1.5.5. Tableau des données consolidées (Version BUEMI) 18](#_Toc189646296)

[2.2. Gestion des logs 26](#_Toc189646297)

[2.3. Annexes 27](#_Toc189646298)

# Description générale

|  |  |
| --- | --- |
| **Contexte / Objectifs** | Consolidation des données extraites des différentes sources CAO (SME, REVIT, NOEE, OCARINA)  Affichage des données consolidées dans une IHM.  Modification des propriétés code client, code fournisseur et de tenue chocs par un utilisateur |
| **RTM** |  |
| **Cartographie Applicative** | ??? |
| **Jalons Navire d’utilisation** |  |
| **Disponibilité** | D3 |
| **Acteurs/Habilitations** | Des rôles devront être attribués aux utilisateurs voulant se connecter à l’application en fonction du projet (Visualisation/Modification). |
| **Accès par des co-réalisateurs distants** | Oui |
| **Traitement batch** | Chaine d’import pour récupérer les données de la CAO et les insérer dans la base et d’export réalimenter les CAO à partir de la base. (Voir SFD spécifiques aux flux de données). |
| **Volumétrie et performance** | S/O. |
| **Impacts sur autres applications du SI** | S/O |
| **Autres …** |  |

|  |  |
| --- | --- |
| **Environnement GitLab** | DEV : Secteur SI Technique\Pole CAO ARM\Double Codification\ |
| **Autres …** |  |

# Description détaillée

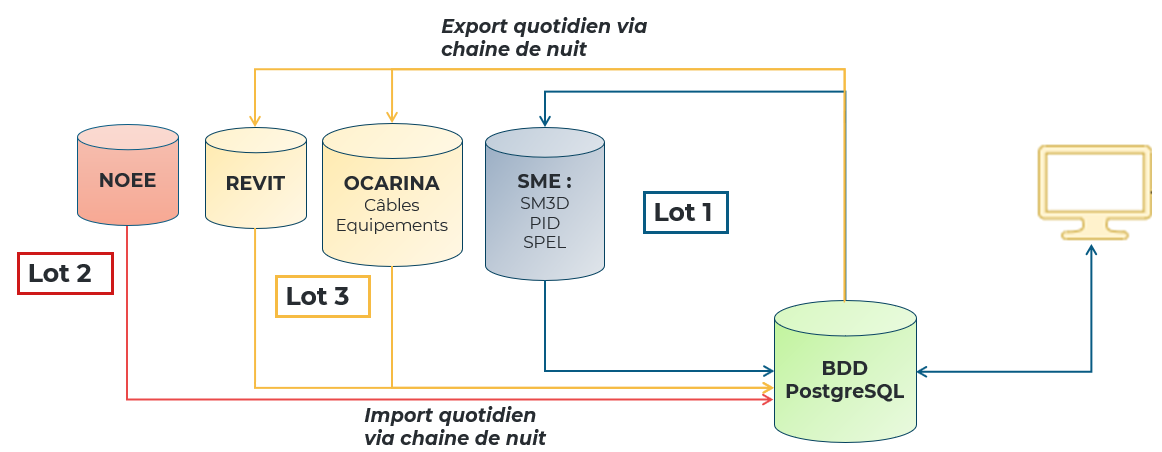
## Description fonctionnelle des Cas d’utilisation

### Description

Dans le cadre du projet de gestion de la double codification, l’objectif est de permettre la centralisation de la gestion de propriétés tel que le code client, le code fournisseur ou la tenue aux chocs au sein d’une application dédiée.

Cette application permettra aux utilisateurs de créer, modifier, supprimer ou uniquement visualiser les correspondances entre les objets nommés présents dans les différentes CAO et ces propriétés.

Les données seront stockées dans une base de données PostgreSQL qui sera mise à jour soit en interactif via l’application soit via des traitements d’imports qui seront exécutés quotidiennement en chaîne de nuit pour les différentes CAO



La double codification reçoit donc des données de différentes sources et doit consolider ces données selon des règles pré-établies. Les données consolidées seront disponibles pour l’application (visualisation, édition).

Toutes les interactions entre l’application et les données de la base double codification se feront via des appels à une API REST qui sera mise à disposition.

L’API sera déployée sous docker swarm. La connexion se fera via un système de login/mot de passe avec vérification via Active Directory pour identifier et contrôler les utilisateurs envoyant des requêtes. L’API logguera dans des fichiers son cycle de vie, ses requêtes et ses erreurs.

### Schéma de la base de données

La base de données postgresql proposera plusieurs schémas et chaque schéma représentera un projet.

Une partie de la base de donnée sera paramétrable par les métiers et l’autre sera alimentée par les différentes sources CAO et l’application.

#### Description du schéma

##### Tables Catalogues

*Table ouvrages sur le schéma public*

|  |  |  |  |
| --- | --- | --- | --- |
| **Colonne** | **Type** | **Nullable** | **Clé** |
| id | Integer | Non | PK |
| ouvrage | Text | Non | UK |
| type | Text | Non | / |
| description | Text | Oui | / |
| image | Text | Oui | / |
| code\_client | Text | Non | / |
| create\_on | DATE | Oui | / |
| is\_active | BOOLEAN | Oui | / |

*Table facteur\_choc sur le schéma du projet :*

|  |  |  |  |
| --- | --- | --- | --- |
| **Colonne** | **Type** | **Nullable** | **Clé** |
| id | Integer | Non | PK |
| value | Text | Oui | UK |

*Table degre\_choc sur le schéma du projet :*

|  |  |  |  |
| --- | --- | --- | --- |
| **Colonne** | **Type** | **Nullable** | **Clé** |
| id | Integer | Non | PK |
| value | Text | Oui | UK |

*Table avec\_plots sur le schéma du projet :*

|  |  |  |  |
| --- | --- | --- | --- |
| **Colonne** | **Type** | **Nullable** | **Clé** |
| id | Integer | Non | PK |
| value | Text | Oui | UK |

*Table avec\_carlingage sur le schéma du projet :*

|  |  |  |  |
| --- | --- | --- | --- |
| **Colonne** | **Type** | **Nullable** | **Clé** |
| id | Integer | Non | PK |
| value | Text | Oui | UK |



Par défaut on affichera, dans l’application, la liste de valeurs dans l’ordre croissant sur la colonne value.

##### Table métier

*Table objects\_from\_cao*

|  |  |  |  |
| --- | --- | --- | --- |
| **Colonne** | **Type** | **Nullable** | **Clé** |
| id | Integer | NON | PK |
| uid | Text | NON | UK |
| source | Text | NON | UK |
| name | Text | NON | / |
| component\_type | Text | OUI | / |
| description | Text | OUI | / |
| trade | Text | OUI | / |
| function | Text | OUI | / |
| lot | Text | OUI | / |
| room | Text | OUI | / |
| code\_client\_object | Text | OUI | / |
| code\_fournisseur | Text | OUI | / |
| facteur\_choc\_id | integer | OUI | FK vers la table facteur\_choc (colonne id) |
| degre\_choc\_id | integer | OUI | FK vers la table degre\_choc (colonne id) |
| avec\_plots\_id | integer | OUI | FK vers la table avec\_plots (colonne id) |
| avec\_carlingage\_id | integer | OUI | FK vers la table avec\_carlingage (colonne id) |
| creation\_date | Timestamp | OUI | / |
| date\_last\_modified | Timestamp | OUI | / |
| date\_last\_modified\_dc | Timestamp | OUI |  |
| status | Text | NON |  |

*Table consolidated\_objects*

|  |  |  |  |
| --- | --- | --- | --- |
| **Colonne** | **Type** | **Nullable** | **Clé** |
| id | Integer | NON | PK |
| source | Text | NON | / |
| name | Text | NON | / |
| component\_type | Text | OUI | / |
| description | Text | OUI | / |
| trade | Text | OUI | / |
| function | Text | OUI | / |
| lot | Text | OUI | / |
| room | Text | OUI | / |
| code\_client\_object | Text | OUI | / |
| code\_fournisseur | Text | OUI | / |
| facteur\_choc\_id | integer | OUI | FK vers la table facteur\_choc (colonne id) |
| degre\_choc\_id | integer | OUI | FK vers la table degre\_choc (colonne id) |
| avec\_plots\_id | integer | OUI | FK vers la table avec\_plots (colonne id) |
| avec\_carlingage\_id | integer | OUI | FK vers la table avec\_carlingage (colonne id) |
| creation\_date | Timestamp | OUI | / |
| date\_last\_modified | Timestamp | OUI | / |
| date\_last\_modified\_dc | Timestamp | OUI |  |
| status | Text | NON | / |

*Table owner\_code\_details*

|  |  |  |  |
| --- | --- | --- | --- |
| **Colonne** | **Type** | **Nullable** | **Clé** |
| id | Integer | NON | PK |
| objconso\_id | Integer | NON | FK sur table consolidated\_objects |
| fieldorder | Text | NON | / |
| fieldvalue | Text | OUI | / |

*Table owner\_code\_properties*

|  |  |  |  |
| --- | --- | --- | --- |
| **Colonne** | **Type** | **Nullable** | **Clé** |
| id | Integer | NON | PK |
| fieldorder | Integer | NON | UK |
| fieldtype | Text | NON | / |
| fieldvalue | Text | OUI | / |
| fieldlabel | Text | OUI | / |

*Table owner\_code\_lov :*

Cette table sera utilisée pour paramétrer les listes de valeurs pouvant être utilisée pour alimenter certaines parties du code client. Le lien entre les tables owner\_code\_lov et owner\_code\_properties se fera respectivement sur les colonnes name et fieldvalue.

|  |  |  |  |
| --- | --- | --- | --- |
| **Colonne** | **Type** | **Nullable** | **Clé** |
| id | Integer | Non | PK |
| name | Text | Non | / |
| value | Text | Non | / |
| description | Text | Oui | / |

*Table property\_consolidation\_param :*

|  |  |  |  |
| --- | --- | --- | --- |
| **Colonne** | **Type** | **Nullable** | **Clé** |
| id | Integer | NON | PK |
| property\_name | Text | NON | UK |
| source\_priority | Text | OUI | / |
| display\_mode | Text | OUI | / |

### API Double Codification

#### Généralités et paramétrages

Toute application voulant intéragir avec les données de la base double codification ne devra pas interroger directement la base de données. Une API sera mise à disposition pour communiquer avec la base de données.

L’API sera déployée sous docker swarm. L’authentification se fera via la saisie d’un login et mot passe sur une page d’accueil et vérification des groupes auxquels l’utilisateur est associé dans Active Directory. L’API logguera dans des fichiers son cycle de vie, ses requêtes et ses erreurs.

#### Descriptions des contrôleurs

|  |  |
| --- | --- |
| **RG** | **Description** |
| RG-00-01 | **ConsolidateDatas [POST] :**   * Prend en paramètre un ouvrage. * Effectue la consolidation des données afin d’alimenter la table consolidated\_objects à partir de la table objects\_from\_cao puis mets à jour la table objects\_from\_cao si nécessaire. (Voir [Job de Consolidation des données](#_Job_de_Consolidation)). |
| RG-00-02 | **GetConsolidatedDatas [GET] :**   * Prend en paramètre un ouvrage et un nombre de page. * Permet de récupérer les objets consolidées (table consolidated\_objects) et d’alimenter les différents champs constituants le code client (table owner\_code\_details). * Utiliser pour alimenter le tableau des données consolidées de l’écran principal. |
| RG-00-03 | **GetCatalogDatas [GET]:**   * Prend en paramètre un ouvrage. * Utiliser pour alimenter les listes de valeurs du tableau de l’écran principal pour les colonnes Facteur choc, Degré choc, Avec plots, Avec carlingage et des champs constituants le code client associés à une liste de valeur. * Non exposé => Fonctionnalité interne à l’application. |
| RG-00-04 | **CheckOwnerCodeUnicity [GET]:**   * Prend en paramètre un ouvrage et un code client. * Vérifie l’unicité d’un code client en cours de création ou modification. * Non exposé => Fonctionnalité interne à l’application. |
| RG-00-05 | **CheckNameUnicity [GET]:**   * Prend en paramètre un ouvrage et un repère fonctionnel. * Vérifie l’unicité du repère fonctionnel renseigné dans le cadre d’une saisie anticipée. * Non exposé => Fonctionnalité interne à l’application. |
| RG-00-06 | **UpdateDatas [POST] :**   * Prend en paramètre un ouvrage, une source et un fichier json**.** * Utiliser pour traiter le fichier json généré par le job d’extraction des données CAO correspondant à la source passée en paramètre et mettre à jour la base de données double codification. * Voir SFD en annexe 2 (RG-02-03, 04 et 05) pour le fonctionnement attendu des créations, modifications et archivages. |
| RG-00-07 | **UpdateDatasFromUser [POST] :**   * Prend en paramètre :   + un ouvrage.   + une liste de repère fonctionnel.   + un type d’action. * Utiliser pour mettre à jour la base de données suite aux créations, modifications et/ou archivages effectuées dans l’application. |
| RG-00-08 | **GetDatasForCAO [Get]:**   * Prend en paramètre un ouvrage, une source et une date de traitement. * Utiliser pour exporter, pour un ouvrage donné, dans un fichier json les objets associés à la source passée en paramètre du job et ayant été modifiés depuis la dernière exécution du traitement pour le couple ouvrage/source. |

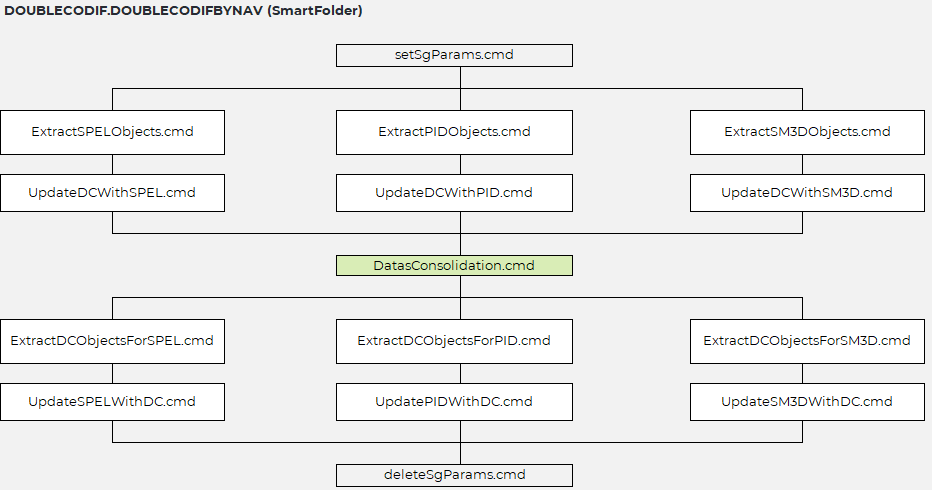
### Job de Consolidation des données

#### Généralités et paramétrages

Les données présentes dans la table objects\_from\_cao n’ayant pas vocation à bouger en dehors de leurs mises à jours quotidienne via chaîne de nuit, dans un objectif de performance la consolidation des données sera effectuée par un job et non dynamiquement à chaque sélection d’un ouvrage ou rafraissement des données dans l’application.

Le job sera nommé DatasConsolidation et sera exécuté à la suite des jobs UpdateDCWithSM3D, UpdateDCWithPID et UpdateDCWithSPEL.

L’enchainement des jobs sera le suivant :



|  |  |
| --- | --- |
| **RG** | **Description** |
| RG-00-08 | Chaque objet présent en base de données dans la table objects\_from\_cao provient d’une seule et unique source et a un identifiant unique issue de cette source (colonne uid).  Un objet consolidé sera créé en prenant comme clé de regroupement le repère fonctionnel (colonne name de la table objets\_from\_cao). C’est via cette clé que les objets de différentes sources représentant l’objet consolidé seront identifiés.  Seuls les objets pour lesquels le statut est différent de D dans la base de données devront être pris en compte pour la consolidation (status != D)  La consolidation sera effectué via un appel au point d’API **ConsolidateDatas**. |
| RG-00-09 | Un même objet consolidé pourra avoir plusieurs source CAO en base de données et plusieurs occurrences pour une même source.  Chaque occurrence, d’un objet consolidé, aura un identifiant unique issu de sa source CAO (colonne uid de la table objets\_from\_cao).  La consolidation **pour chaque propriété** se fera suivant des règles de priorités et d’affichage définies par MOS et décrites dans la table de paramétrage property\_consolidation\_param.  Cette table sera constituée des colonnes suivantes :   * property\_name : nom de la propriété dans les tables objects\_from\_cao et consolidated\_objects. * source\_priority : indique l’ordre de priorité dans lequel les sources doivent être prises en compte afin d’alimenter la colonne associée à la propriété. Les différentes sources seront séparés par un « ; ». * display\_mode : indique la manière suivant laquelle les données seront consolidées et présentées à l’utilisateur dans l’application.   *Exemple d’alimentation de la table :*   |  |  |  |  | | --- | --- | --- | --- | | **id** | **property\_name** | **source\_priority** | **display\_mode** | | 1 | source |  | All | | 2 | component\_type | SM3D;PID;SPEL;USER | First | | 3 | description | SM3D;PID;SPEL;USER | First | | 4 | trade | SM3D;PID;SPEL;USER | First | | 5 | function | SM3D;PID;SPEL;USER | First | | 6 | lot | SM3D;PID;SPEL;USER | All | | 7 | room | SM3D;PID;SPEL;USER | All | | 8 | code\_client\_object | SM3D;PID;SPEL;USER | First | | 9 | code\_fournisseur | SPEL;USER | No | | 10 | facteur\_choc\_id | SPEL;USER | First | | 11 | degre\_choc\_id | SPEL;USER | First | | 12 | avec\_plots\_id | SPEL;USER | First | | 13 | avec\_carlingage\_id | SPEL;USER | First |   La consolidation des données sera effectuée pour chaque propriété en fonction de la valeur des colonnes source\_priority et display\_mode.  La colonne display\_mode pourra prendre pour valeur **First, All ou No**.  Les colonnes ayant une valeur de display\_mode égale à **No** ne seront :   * Pas alimentée dans la table consolidated\_objects. * Pas affichée dans l’IHM. * Pas exportée dans le json de mise à jour des CAO. * Le fonctionnement de la consolidation sera le suivant :Si une valeur est présente dans la colonne source\_priority pour la propriété à consolider. Alors on regardera pour chaque occurrence du repère fonctionnel en cours de traitement si celui-ci dispose dans la table objects\_from\_cao de la source définie comme prioritaire.   + Si oui,     - Si la colonne display\_mode est égale à **First**, alors :       * S’il existe une seule occurrence pour le repère fonctionnel associé à la source prioritaire, alors la propriété consolidée sera égale à la valeur de cette occurrence.       * S’il existe plusieurs occurrences pour le repère fonctionnel associé à la source prioritaire **avec des valeurs différentes** pour la propriété en cours de consolidation, alors la propriété consolidée sera égale à la valeur associé à l’id le plus petit parmi les occurences.     - Si la colonne display est égale à **All**, alors :       * S’il existe une seule occurrence pour le repère fonctionnel associé à la source prioritaire, alors la propriété consolidée sera égale à la valeur de cette occurrence.       * S’il existe plusieurs occurrences pour le repère fonctionnel associé à la source prioritaire **avec des valeurs différentes** pour la propriété en cours de consolidation, alors la propriété consolidée sera égale la concaténation des valeurs distinctes triées par ordre alphabétique.   + Si non,     - Si la colonne display\_mode est renseignée et égale à **First** alors :       * S’il existe une seule occurrence pour le repère fonctionnel, alors la propriété consolidée sera égale à la valeur de cette occurrence.       * S’il existe plusieurs occurrences pour le repère fonctionnel **avec des valeurs différentes** pour la propriété en cours de consolidation, alors la propriété consolidée sera égale à la valeur associé à l’id le plus petit parmi les occurences.     - Si la colonne display\_mode est renseignée et égale à **All** alors :       * S’il existe une seule occurrence pour le repère fonctionnel, alors la propriété consolidée sera égale à la valeur de cette occurrence.       * S’il existe plusieurs occrrence pour le repère fonctionnel associé **avec des valeurs différentes** pour la propriété en cours de consolidation, alors la propriété consolidée sera égale la concaténation des valeurs distinctes triées par ordre alphabétique.   Si une propriété n’est pas présente dans la table de paramétrage alors il n’y aura pas de consolidation pour cette propriété à l’exception des colonnes :   * creation\_date qui ne sera jamais modifiée une fois l’objet consolidé créé. * date\_last\_modified qui sera mis à jour si une des propriétés de l’objet consolidé est modifiée. * date\_last\_modified\_dc qui sera mis à jour si celle-ci est antérieure à la date\_last\_modified\_dc présente sur une occurrence active (status != D) dans la table objects\_from\_cao. Ceci signifiera que lors de l’import des données depuis les CAO une incohérence a été détectée entre la valeur issue de la CAO et celle présente dans la double codification pour une ou des propriétés pilotées par l’application (code fournisseurs, facteur de choc, degré de choc, avec plots et/ou avec carlingage).   Si pour une propriété présente dans la table de paramétrage, les sources devant être utilisées pour la consolidation n’existent pas pour le repère fonctionnel en cours de traitement alors la valeur consolidée sera vide.  *Exemple :* repère fonctionnel avec une source PID et pour la propriété lot consolidation définie comme devant se baser uniquement sur les sources SM3D et SPEL => Lot consolidé = vide.  Si pour une propriété la valeur associée à l’occurrence devant être utilisée pour la consolidation est égale à vide alors la priorité consolidée sera également égale à vide. On ne regardera pas si une valeur est renseignée sur une autre source. |
| RG-00-10 | **Mise à jour de la table consolidated\_objects :**  La table pourra être mise à jour de 3 manières différentes en fonction du nombre d’occurrence(s) existante(s) pour le repère fonctionnel :   1. ***1 seule ligne pour un repère fonctionnel :***    1. **Création  :**       1. Aucune ligne présente dans la table consolidated\_objects pour le repère fonctionnel en cours de traitement => Création d’une nouvelle ligne avec alimentation des colonnes :          1. creation\_date          2. date\_last\_modified          3. status = A (Add).    2. **Modification :**       1. Un repère fonctionnel présent dans la table objects\_from\_cao, avec un statut différent de D (colonne status), et présent dans la table consolidated\_objects, avec un statut différent de D (colonne status), entrainera le recalcul de l’objet consolidé afin de prendre en compte les modifications. Mise à jour des colonnes :          1. date\_last\_modified          2. status = M (Modify)          3. date\_last\_modified\_dc si mise à jour d’une propriété pilotée par l’application    3. **Suppression :**       1. Un repère fonctionnel présent dans la table objects\_from\_cao, avec un statut égale à D, et présent dans la table consolidated\_objects, avec un statut différent de D, entrainera l’archivage de l’objet consolidé. Mise à jour des colonnes :          1. date\_last\_modified          2. status = D (Delete) 2. ***Plusieurs lignes pour un repère fonctionnel :***    1. **Création  :**       1. Aucune ligne, présente dans la table consolidated\_objects pour le repère fonctionnel en cours de traitement => Création d’une nouvelle ligne avec alimentation des colonnes :          1. creation\_date.          2. date\_last\_modified          3. status = A.    2. **Modification :**       1. Le repère fonctionnel, en cours de traitement, est présent dans la table objects\_from\_cao, avec un statut différent de D, et présent dans la table consolidated\_objects, avec un statut différent de D, alors on recalcul l’objet consolidé afin de prendre en compte :          1. L’ajout d’une nouvelle ligne sur une source déjà existante ou sur une nouvelle source.          2. La suppression d’une ligne sur une source déjà existante avec de multiples occurrences ou une unique occurence.          3. La modification des propriétés (hors propriétés pilotées par l’application).          4. La modification du paramétrage des sources prioritaires et/ou de la prise en compte des valeurs (unique ou concaténée).       2. Mise à jour des colonnes :          1. date\_last\_modified          2. status = M.          3. date\_last\_modified\_dc si mise à jour d’une propriété pilotée par l’application    3. **Suppression :**       1. Toutes les lignes associées au repère fonctionnel, présentes dans la table objects\_from\_cao, sont archivées (status = D) alors que l’objet consolidé ne l’est pas (statut != D) => Archivage de l’objet consolidé (status = D).       2. Mise à jour des colonnes :          1. date\_last\_modified          2. status = D. |
| RG-00-11 | **Mise à jour de la table objects\_from\_cao :**  Suite à la consolidation des données dans la table consolidated\_objects, si des modifications ont été effectuées sur les données pilotées par l’application il faudra les propager sur les n occurrences associées à chaque objet consolidé dans la table objects\_from\_cao.  *Exemple :*   1. Import depuis SM3D du repère REP01 avec code\_fournisseur = FRN1 2. Import depuis PID du repère REP01 avec code fournisseur = vide 3. Consolidation du repère REP01 avec source prioritaire pour le code fournisseur = SM3D => code\_fournisseur objet consolidé = FRN1 4. Mise à jour dans la table objects\_from\_cao de l’occurrence PID pour avec code\_fournisseur = FRN1 5. Export vers PID du repère REP01 avec code fournisseur = FRN1 |

### Application Double Codification

Afin de permettre aux utilisateurs d’intéragir avec les données de la base double codification, une IHM leur sera mise à disposition.

#### Généralités et paramétrages

L’application sera développée en Django Python et sera paramétrée avec un fichier de configuration donnant les information suivantes :

* Adresse de base de l’API
* Le niveau de log

Chaque session utilisateur sera logguer sur le serveur.

#### Fonctionnalitées principales

Les fonctionnalités principales de l’application seront la visualisation, la création et la modification de données. Afin de remplir cet objectif, plusieurs autres fonctionnalités seront ajoutées en support tel que le filtrage, la saisie anticipée ou l’aide à la saisie du code client.

#### Accès à l’application

|  |  |
| --- | --- |
| **RG** | **Description** |
| RG-01-01 | Seules les personnes autorisées sur un ouvrage pourront accéder à celui-ci via l’application.  2 groupes seront définis dans l’annuaire pour chaque ouvrage et seront associés dans l’application à des permissions stockées dans le schéma public de la base de données :   * DC\_<NAV>\_VIEWER => associé à un accès en visualisation. * DC\_<NAV>\_EDITOR => associé à un accès en édition.   1 groupe sera également défini pour un accès multi-ouvrage en édition :   * DC\_SUPPORT   La connexion se fera via une page d’accueil permettant la saisie d’un login et mot de passe. Lors de la connexion à l’application on devrait vérifier via l’AD à quel(s) groupe(s) appartient l’utilisateur afin de lui afficher la liste des ouvrages auxquels il a accès avec le bon niveau de droits. |
| RG-01-02 | Après connexion à l’application, la page d’accueil affichera uniquement les ouvrages pour lesquels l’utilisateur dispose des autorisations sous forme la forme d’une vignette contenant :   * Le nom de l’ouvrage * Une description (optionnelle) * Une image de taille 300\*150 qui sera fournie par MOS.   Ces informations seront paramétrées dans la table ouvrages sur le schéma public de la base double codification.  Les vignettes auront une bordure de couleurs différentes en fonction des droits utilisateur sur l’ouvrage (exemple à valider : vert => accès global, jaune => visualisation) |
| RG-01-03 | La sélection d’un projet sur la page d’accueil entraine automatiquement le basculement sur l’affichage de l’écran principal et du tableau des données consolidées.  Pour revenir à la page d’accueil et sélectionné un nouvel ouvrage il faudra cliquer sur le bouton home/accueil. |
| RG-01-04 | **Accès en mode visualisation :**  Pour les utilisateurs ne disposant sur un projet que de droit d’accès en visualisation, les données seront grisées dans les cellules.  Les boutons permettant les actions suivantes seront inactifs :   * Création d’une nouvelle ligne. * Archivage. * Validation des données.   Seules les actions suivantes seront possibles :   * Filtrage. * Tri. * Réinitialisation des filtres. * Retour au choix du projet. |

#### Ecran principal de l’application

|  |  |
| --- | --- |
| **RG** | **Description** |
| RG-02-01 | L’écran principal sera composé d’une en-tête (1), du tableau des données consolidées (2) et d’un bas de page (3). |
| RG-02-02 | L’en-tête (1) contiendra :   * Le logo Chantiers de l’Atlantique. * Un texte indiquant le projet sélectionné. * Un bouton de validation (sauvegarder des modifications). * Un bouton de rafraichissement des données. * Un bouton pour réinitialiser tous les filtres appliqués dans le tableau. * Un bouton pour insérer une ligne pour saisie anticipée. * Un bouton pour archiver les données. * Un bouton home/accueil pour retourner à la page de sélection de l’ouvrage |
| RG-02-03 | Le tableau (2) contient les données consolidées. |
| RG-02-04 | Le base de page (3) sera composé :   * Un texte indicant le nombre d’objets total. * Le numéro de la page active et le nombre total de pages. |

#### Tableau des données consolidées (Version BUEMI)

|  |  |
| --- | --- |
| **RG** | **Description** |
| RG-03-01 | L’application affichera à l’utilisateur un tableau listant l’ensemble des objets consolidés présent dans la table consolidated\_objects sous la forme suivante :    Afin de faciliter la visualisation, une ligne sur 2 sera coloriée en gris clair (à valider avec MOS).  Le tableau disposera d’ascenceurs horizontaux et verticaux pour permettre de visualiser l’ensemble des données.  Par défaut, les objets seront triés par ordre alphabétique de repère fonctionnel. Il sera possible d’effectuer un tri sur chaque colonne du tableau. Il ne sera pas possible de combiner plusieurs tris.  Les 2 premières colonnes devront toujours être visibles ainsi que l’entête des colonnes. (A définir et valider). Faut-il prévoir de faire une fit sur les colonnes ou avoir une taille fixe ?  Dans l’application, le tableau affichera les colonnes suivantes :   |  |  |  | | --- | --- | --- | | **Nom de colonne** | **Modifiable** | **Type** | | Repère fonctionnel | Non | Texte | | Source | Non | Texte | | Description | Non | Texte | | Type de composant | Non | Texte | | Métier | Non | Texte | | Fonction | Non | Texte | | Lot | Non | Texte | | Local | Non | Texte | | Code client | Oui | Texte | | Code fournisseur | Oui | Texte | | Facteur de choc | Oui | Liste de valeurs | | Degré de choc | Oui | Liste de valeurs | | Avec plots | Oui | Liste de valeurs | | Avec carlingage | Oui | Liste de valeurs | | Date de création | Non (MaJ suite validation) | DateTime | | Date de modification | Non (MaJ suite validation) | DateTime |   Les colonnes pour lesquelles les valeurs sont non modifiables par l’utitisaleur auront une police d’écriture en gris foncé (à valider en fonction de la couleur choisie pour la colorisation des lignes). |
| RG-03-02 | **Alimentation du tableau :**  Les colonnes du tableau seront alimentées de la manière suivante :   * Repère fonctionnel = propriété **name** de la table consolidated\_objects. * Source = propriété **source** de la table consolidated\_objects. * Description = propriété **description** de la table consolidated\_objects. * Type de composant = propriété **component\_type** de la table consolidated\_objects. * Métier = propriété **trade** de la table consolidated\_objects. * Fonction = propriété **function** de la table consolidated\_objects. * Lot = propriété **lot** de la table consolidated\_objects. * Local = propriété **room** de la table consolidated\_objects. * Code client = Voir RG-03-03. * Code fournisseur = propriété **code\_fournisseur** de la table consolidated\_objects. * Facteur de choc = propriété value de la table **facteur\_choc** en parcourant la relationconsolidated\_objects.facteur\_choc\_id = facteur\_choc.id. * Degré de choc = propriété value de la table **degre\_choc** en parcourant la relation consolidated\_objects.degre\_choc\_id = degre\_choc.id. * Avec plots = propriété value de la table **avec\_plots** en parcourant la relation consolidated\_objects.avec\_plots\_id = avec\_plots.id. * Avec carlingage = propriété value de la table **avec\_carlingage** en parcourant la relation consolidated\_objects.avec\_carlingage\_id = avec\_carlingage.id. * Date de création = propriété **creation\_date** de la table consolidated\_objects. * Date de modification = date la plus récente entre les propriétés **date\_last\_modified** et **date\_last\_modified\_dc** de la table consolidated\_objects. |
| RG-03-03 | **Gestion du code client :**  Le code client devra répondre à des normes de nommages différentes en fonction du type de projet (BUEMI ou PANG).  Nous décrirons dans le cadre du lot 1 uniquement la saisie d’un code client pour un projet de type BUEMI.  Le masque de saisie sera le suivant (il pourra être amené à évoluer) :    Afin de rendre le masque de saisie evolutif en fonction du type de projet et des possibles modifications des normes de nommages, la cellule contenant le code client se présentera sous la forme d’une cellule encapsulant les n champs constituants le code client.  Une table de paramétrage à l’ouvrage nommée owner\_code\_properties et constituée des colonnes :   * Fieldorder. Indique la position du champ dans la cellule. * Fieldtype. Indique le type d’alimentation pour un champ. Ce type pourra prendre les valeurs suivantes :   + 1 : Fixe   + 2 : LOV   + 3 : Regex. (Saisie libre mais validée par une Regex) * Fieldvalue. En fonction de la valeur de la colonne Fieldtype, la valeur correspondra à :   + Si Fieldtype = Fixe : valeur fixe non modifiable   + Si Fieldtype = LOV :     - les valeurs seront à récupérer dans la table de paramétrage owner\_code\_lov. Le lien entre les tables owner\_code\_lov et owner\_code\_properties se fera respectivement sur les colonnes name et fieldvalue.     - Dans l’IHM, les listes de valeurs seront affichées à l’utilisateur en concaténant les propriétés value et description de la table owner\_code\_lov de la manière suivante : value – description.   Il sera possible, comme pour les filtres, de restreindre le contenu de la liste via saisie.  La valeur stockée dans le code client correspondra uniquement à la propriété value.   * + Si Fieldtype = Regex : la colonne contiendra la Regex à respecter. * Un label qui servira de tooltip la propriété à laquelle le champ correspond.   Il ne pourra pas y avoir de liste de valeurs hiérarchique.  Lorsque l’on cliquera dans la cellule, celle-ci prendra par exemple (sur le modèle du masque BUEMI) la forme suivante :    Pour chaque repère fonctionnel consolidé, si un code client existe alors lors de l’alimentation du tableau il faudra récupérer les valeurs des champs dans la table owner\_code\_details via la relation consolidated\_objects.id=owner\_code\_details.nameid.  La table owner\_code\_details contiendra les colonnes suivantes :   * objconso\_id : clé de liaison avec la table des objets consolidés. * fieldorder : indiquant la position de la propriété dans les champs constituants le code client. * fieldvalue : la valeur de la propriété.   Le code client ne devra pas :   * se terminer par un espace. Cas où :   + La partie P2 n’est pas renseignée.   + Les parties P1 et P2 ne sont pas renseignées. * se terminer par un tiret + un espace => cas où les parties P1 **et** P2 ne sont pas renseignées. * avoir un tiret suivi d’un espace => cas où la partie P1 n’est pas renseignée.   Toutes les créations ou modifications de codes clients, une fois les contrôles de saisies et d’unicité passés (Voir RG-03-04), devront être propagés dans la table owner\_code\_details en plus de la table consolidated\_objects qui elle stockera uniquement la valeur complète du code client dans une optique d’alimentation de la table objects\_from\_cao et d’utilisation par les flux de mises à jour des CAO.  Les colonnes suivantes de la table consolidated\_objects seront également mises à jour :   * date\_last\_modified\_dc : date de dernière mise à jour d’une propriété pilotée par l’application double codification. * status = M.   Ces mises à jours seront propagées dans la table objects\_from\_cao. |
| RG-03-04 | **Contrôles sur le code client :**  Il existera pour la propriété code client 2 types de contrôles :   1. Un contrôle de saisie qui sera fait dynamiquement pour chacun des champs constituants le code client. Si celui-ci est KO alors le champ sera mis en surbrillance rouge. 2. Un contrôle d’unicité qui sera effectué lorsque l’utilisateur sortira de la cellule via un appel au point d’API CheckOwnerCodeUnicity. Si le contrôle est KO le champ sera en surbrillance rouge.   Si l’utilisateur sort de la cellule, alors le code client est réaffiché sous une forme unifiée et non éclatée. Si un des champs le constituant est en erreur alors la cellule est dans sa globalité en surbrillance rouge. Si l’utilisateur repasse à la forme éclatée, seuls les champs en erreurs sont en surbrillance rouge.  *Exemples :*   * Forme « unifiée » OK :      * Forme « éclatée » OK :      * Forme unifiée KO pour unicité non respectée :      * Forme éclatée KO si n ou n champs ne respectent pas le type attendu : |
| RG-03-05 | **Filtrage des données :**  Les en-tête de chaque colonne du tableau posséderont un filtre propre au type de données représentées dans la colonne.  Le filtrage sur les colonnes de type Texte ou Liste de valeurs aura 4 fonctions de filtre disponibles :   * Liste : affiche uniquement la liste des valeurs cochées. * Commence par : affiche tous les éléments commençant par le texte indiqué. * Contient : affiche tous les éléments contenant le texte indiqué. * Ne contient pas : affiche tous les éléments ne contenant pas le texte indiqué   Les filtres se présenteront sous la forme suivante :    Le filtrage sur les colonnes de type DateTime s’appliquera à tous les champs DateTime et il aura 4 fonctions :   * Liste : affiche uniquement la liste des valeurs cochées * Supérieure ou égale : affiche tous les éléments dont la valeur est supérieure ou égale à la valeur saisie. * Inférieure ou égale : affiche tous les éléments dont la valeur est inférieure ou égale à la valeur saisie. * Compris entre : affiche tous les éléments dont la valeur est comprise entre les deux valeurs saisies.   Si une valeur absente d’une liste affichée dans un filtre est saisie ou sélectionnée par un utilisateur alors celle-ci doit pouvoir être utilisée dans le filtre dynamiquement.  Il sera possible de filtrer sur la valeur vide afin, par exemple, d’afficher les repères fonctionnels n’ayant pas de code fournisseur renseigné.  Un bouton permettant de réinitialiser l’ensemble des filtres sera disponible. |
| RG-03-06 | Il sera possible de sélectionner plusieurs cellules d’une même colonne pour y appliquer la même valeur. La sélection se fera soit à la souris soit avec la touche shift. Ceci sera également valable pour la multisaisie des champs constituants le code client. |
| RG-03-07 | Pour les cellules texte, il suffira de sélectionner les cellules voulues puis taper la valeur désirée et appuyer sur entrée ou cliquer simplement ailleurs pour valider les cellules. |
| RG-03-08 | Pour les cellules d’énumération (liste de valeurs), après sélection il faudra appuyer sur shift pour entrer en édition de cellule et sélectionner une valeur dans la liste déroulante. Il suffira ensuite d’appuyer sur entrée ou cliquer ailleurs pour valider les cellules. |
| RG-03-09 | **Gestion de la mise à jour des données :**  Dans l’application, seules les propriétés pilotées par celle-ci seront accessibles en modification à l’exception des lignes créées en anticipation et avec une source USER (Voir RG-03-10). Il s’agira des colonnes :   * Code client : champ(s) constituant(s) le code client. * Code fournisseur * Facteur de choc * Degré de choc * Avec plots * Avec carlingage   La colonne **Code fournisseur** se présentera sous la forme d’une cellule en saisie libre.  La colonne **Facteur de choc** se présentera sous la forme d’une liste de valeurs. Ces valeurs seront issues de la table facteur\_choc (voir MDD). Il ne sera pas possible de saisir de nouvelle valeur pour cette liste depuis l’application.  La colonne **Degré de choc** se présentera sous la forme d’une liste de valeurs. Ces valeurs seront issues de la table degre\_choc (voir MDD). Il ne sera pas possible de saisir de nouvelle valeur pour cette liste depuis l’application.  La colonne **Avec plots** se présentera sous la forme d’une liste de valeurs. Ces valeurs seront issues de la table avec\_plots (voir MDD). Il ne sera pas possible de saisir de nouvelle valeur pour cette liste depuis l’application.  La colonne **Avec carlingage** se présentera sous la forme d’une liste de valeurs. Ces valeurs seront issues de la table avec\_carlingage (voir MDD). Il ne sera pas possible de saisir de nouvelle valeur pour cette liste depuis l’application.  Pour ces 2 propriétés, il sera possible de remettre la valeur à vide via la liste de valeurs proposées. |
| RG-03-10 | **Création anticipée :**  Il sera possible en cliquant sur un bouton Add New Line (+) de créer une nouvelle ligne dans une optique de saisie anticipée des propriétés pour un repère fonctionnel non remonté depuis au moins une source CAO.  La nouvelle ligne apparaitra en 1er dans le tableau et sera initialisée avec la valeur USER, **non modifiable**, dans la colonne Source. En plus de la colonne Source, les colonnes Code Date de création et Date de modification seront grisées et non modifiables.  En revanche, toutes les autres colonnes seront accessibles et modifiables. Il n’y aura pas de contrôles effectués sur les valeurs saisies dans les colonnes Description, Type de composant, Métier, Fonction, Lot et Local.  **L’utilisateur devra respectée 3 conditions pour que la ligne soit valide :**   1. Les propriétés minimales suivantes seront renseignées :    * Repère fonctionnel.   Et   * + Une des propriétés pilotées par l’application :     - Code client     - Code fournisseur     - Facteur de choc     - Sur Plots  1. Le repère fonctionnel renseigné ne devra pas exister dans la base de donnée (appel au point d’API CheckNameUnicity) :    * Repère fonctionnel inexistant.   Ou   * + Repère fonctionnel existant mais avec archived\_date non null pour toutes les sources auxquelles il est associée.  1. Si le code client est saisi, son unicité devra être contrôlée (appel au point d’API CheckOwnerCodeUnicity).   Lorsqu’une nouvelle ligne est en cours de création, il n’est pas possible de cliquer sur une autre ligne sans avoir valider ou annuler la création en cours.  La validation, dans l’application, se fera en cliquant sur le bouton de validation (coche verte - appel au point d’API UpdateDatasFromUser).  Si les 3 conditions précitées ne sont pas remplies, la validation se terminera en erreur et un message devra indiquer à l’utilisateur le ou les raisons empêchant celle-ci.  Les colonnes d’audit prendront pour les lignes créées en anticipation les valeurs :   * creation\_date : date de 1ère validation de la ligne * status = A   Si la ligne est ensuite modifiée alors :   * creation\_date n’est pas mise à jour. * date\_last\_modified = date de validation * status = M |
| RG-03-11 | **Archivage des données :**  Depuis l’application, il sera possible, en cliquant sur le bouton d’archivage (corbeille), de flagguer comme archivés :   * Un objet consolidé ayant comme unique source USER. En BDD les actions à réaliser seront les suivantes :   + Dans la table consolidated\_objects, l’objet passera au statut D et la prioriété date\_last\_modified sera mise à jour.   + Dans la table objects\_from\_cao, l’objet passera au statut D et la prioriété date\_last\_modified sera mise à jour. * Un objet consolidé ayant une occurrence active dans la table objects\_from\_cao associée à la source USER. Les actions à réaliser seront les suivantes :   + Dans la table objects\_from\_cao, l’occurrence associée à la source USER sera passera au statut D et la prioriété date\_last\_modified sera mise à jour.   + La consolidation devra être reeffectué uniquement pour ce repère fonctionnel.   + Si des modifications ont lieu sur les propriétés pilotées par l’application lors de la consolidation alors :     - Dans la table consolidated\_objects, l’objet passera au statut M et la prioriété date\_last\_modified\_dc sera mise à jour.     - Dans la table objects\_from\_cao, les modifications effectuées seront propagées sur toutes les occurrences ayant des valeurs différentes par rapport à celle de l’objet consolidé.   Les objets ne subiront pas de suppression réelle en base de données. Ils seront archivés et la propriété date\_last\_modified sera mise à jour ainsi que et la colonne status qui sera égale à D. Les objets seront alors considérés comme « supprimés » et ne seront plus pris en compte pour alimenter le tableau.  L’archivage sera effectué dès le clic sur le bouton d’archivage via un appel au point d’API UpdateDatasFromUser. Il ne faudra pas en plus cliquer sur le bouton de validation des modifications.  L’API prendra en entrée la liste des repères fonctionnels correspondants aux lignes sélectionnées par l’utilisateur. Si une suppression n’est pas possible alors un message sera affichée à l’utilisateur.  Si plusieurs lignes sont à supprimer simultannément et qu’une des suppression se termine en erreur alors aucune suppression ne sera effectuée et un message sera affiché à l’utilisateur. |
| RG-03-12 | **Validation des modifications :**  Le bouton de validation (coche verte) est grisé par défaut (non accessible). Dès lors qu’une valeur est modifiée (création/modification) et valide, le bouton devient accessible et il faudra cliquer dessus pour valider toutes les modification apportées. Un appel au point d’API UpdateDatasFromUser sera effectuée afin de mettre à jour les données dans la base de données Postgre SQL.  Même si dans l’application une seule ligne consolidée sera affichée à l’utilisateur, pour un repère fonctionnel, si des modifications sont faites alors celles-ci devront être propagées à tous les objets non archivés ayant le même repère fonctionnel (colonne name) quelque soit leur source dans la table objects\_from\_cao.  Si une erreur se produit lors de la mise à jour d’une ligne, celle-ci sera colorisée en rouge et un message indiquant la cause de l’erreur sera affichée.  Si la validation concerne plusieurs lignes, aucune validation ne sera effectuée si une erreur est rencontrée.  Si des modifications (création/modification/archivage) ont été effectués par un utilisateur dans l’application mais n’ont pas encoré été validées :   * Si l’utilisateur souhaite :   + Changer de projet (retour à la page d’accueil).   + Rafraichir le tableau des données consolidées.   + Fermer l’application.   Alors un message lui indiquant que des modifications en cours n’ont pas été validées et qu’elles seront perdues si il souhaite poursuivre. L’utilisateur aura le choix entre :   * + Cliquer sur OK : perte des modifications non validées.   + Cliquer sur Cancel : retour sur l’écran actuel. |
| RG-03-13 | **Gestion des accès concurrents :**  L’IHM gérera la modification par d’autres utilisateurs ayant une session en cours sur le même scope. Lorsqu’un utilisateur A modifiera la valeur d’une ou de plusieurs propriétés d’un objet alors l’utilisateur B sera notifié dans le tableau que cet objet a été modifié. La ligne correspondant à l’objet sera en surbrillance rouge.  Pour récupérer les valeurs modifiées par un autre utilisateur, il faudra rafraichir via le bouton Refresh (Appel point d’API GetConsolidatedDatas). |

## Gestion des logs

Chaque ligne du log sera formatée comme suit :

DATE-HEURE;TYPE;MESSAGE (ou type peut avoir comme valeur INFO/WARNING/ERROR)

Chaque étape du traitement sera loguée dans un fichier log

Les erreurs techniques (accès à un fichier, commit, …) seront également loguées dans ce fichier.

Il faut utiliser la librairie de log.

Le fichier de log sera dans le dossier log du répertoire du traitement et le log sera nommé avec le nom du job suivi d’un scheduling group s’il y a lieu, du navire et la date.

En complément de ce log, les erreurs techniques et fonctionnelles du traitement seront gérées dans SAO. Les warnings seront traités en tant qu’erreurs fonctionnelles.

## Annexes

**Références**

|  |  |  |  |
| --- | --- | --- | --- |
| N° | Nom du document | Contenu | Lien ALFRESCO |
| 1 | EB GESTION DE LA CODIFICATION CLIENT V2024 | Expression de besoin. | [EB GESTION DE LA CODIFICATION CLIENT V2024.docx](https://share.chantiers-atlantique.com/share/proxy/alfresco/slingshot/node/content/workspace/SpacesStore/a6a889fd-7f35-48df-85cb-d254dc9f4e3e/EB%2520GESTION%2520DE%2520LA%2520CODIFICATION%2520CLIENT%2520V2024.docx?a=true) |
| 2 | SFD - Double Codification - Flux SM3D to DC | SFD | [SFD - Double Codification - Flux SM3D to DC.docx](https://share.chantiers-atlantique.com/share/proxy/alfresco/slingshot/node/content/workspace/SpacesStore/f26c7c1e-0b38-482a-974a-6f566d5bef62/SFD%2520-%2520Double%2520Codification%2520-%2520Flux%2520SM3D%2520to%2520DC.docx?a=true) |
| 3 | SFD - Double Codification - Flux SPEL to DC.docx | SFD | [SFD – Double Codification – Flux SPEL to DC.docx](https://share.chantiers-atlantique.com/share/proxy/alfresco/slingshot/node/content/workspace/SpacesStore/d4ce8c18-57c7-4f3f-b420-a0647acc24d8/SFD%2520-%2520Double%2520Codification%2520-%2520Flux%2520SPEL%2520to%2520DC.docx?a=true) |
| 4 | SFD - Double Codification - Flux PID to DC.docx | SFD | [SFD - Double Codification - Flux PID to DC.docx](https://share.chantiers-atlantique.com/share/proxy/alfresco/slingshot/node/content/workspace/SpacesStore/6783b95a-7905-4f59-8643-c1548cbc6647/SFD%2520-%2520Double%2520Codification%2520-%2520Flux%2520PID%2520to%2520DC.docx?a=true) |