

SDED

Spécifications Techniques Générales

N° Projet	
Direction émettrice (nom du sponsor)	SDED
Responsable MOA	
Responsable MOE	
Description synthétique	

La Spécification Technique Générale (STG) vient en complément du DAT. Il décrit le cadre référentiel en termes de normes applicables et exigences techniques à prendre en compte dans la conception détaillée et l'implémentation.

Rédaction		
Partie prenante	Prénom Nom	Rédacteur (R) / Participant (P)
MOE	RAF - Gaël COSNARD - Guillaume HAMON - Laurent CHARTIER - Willy LENOIR	
	PORA Samir Benaïcha	
SDAE		
SDPIL		
SDAT		
SDIT		
SDOP (+CNE)		
DPO (<i>Délégué à la Protection des données</i>)		

Historique des versions			
Date	Version	Commentaire	Par
31/05/2021	1.0	Initialisation	Gaël Cosnard
02/07/2021	1.1	Ajout Normes, Socle hawaii, Spec Solution	Gaël Cosnard
30/07/2021	V.2	Extraction pour Projet P1	Laurent Chartier
28/11/2021	1.3	Structure des bundles, màj des normes, std p1, conteneurisation, droits des bases de données, openapi, payload de réponse api,	Gaël Cosnard

		historisation, précisions sur les logs, mocks	
15/12/2021	1.4	Ajout norme et guide pour les tests automatisés Offset de démarrage consumer CDC et plages batch Doc complémentaire SNV2	Gaël Cosnard
16/12/2021	1.5	Cache des API Livraison STD sur git Intégration P1	Gaël Cosnard
28/04/2022	1.6	Mise à jour des normes depuis le référentiel d'exigences Découpage des Bundle Archimed Version de doc Debezium	Gaël Cosnard
03/06/2022	1.7	Mise à jour Exigences non fonctionnelles Infos Kafka Bundles Métriques Livraisons PFS Entrants domaines communication	
01/07/2022	1.8	Ajout principes de conception IHM Principe historisation URL dépôts fullstack Particularités PFS dev Documents applicables Frontend & tests	Gaël Cosnard
07/12/2022	1.9	MàJ Fullstack, Doc applicables, livraisons documentaires, Kafka, BIPE, Evolution debezium (date)	Gaël Cosnard
16/02/2023	1.10		Gaël Cosnard

16/02/2023	1.11	Màj document pour la connexion sécurisée à Kafka et aux topics	Khansaâ OUAHMANE
13/06/2023	2.0	PostGREST ProxyTransfert Automator /CJP Subchart postgres Chart namespaces Kiosk+elk IHM +https Lien charte Debezium AKHQ	Gaël Cosnard Willy Lenoir
01/01/2024	3.0	Exigences Non Fonctionnelles Lisibilité du code Bonnes pratiques Logs S3 Suppression compatibilité sgbd oracle	Gaël Cosnard Samir Benaicha
21/05/2024	3.1	Mise à jour des ENF (§2.1) Mise à jour des doc applicables (§2.2) Restructuration Kafka Ajouts Divers	Gaël Cosnard
26/09/2024	3.2	Vu d'ensemble Socles Déplacement de la partie Design Patterns Kafka Livraison	Gaël Cosnard
20/01/2025	3.3	Modification description Header Kafka Workflow Git	Stéphane Martin Gaël Cosnard
25/07/2025	3.4	PFS v2	Gaël Cosnard
19/02/2026	3.5	Mise à jour des ENF et Normes version des lib	Gaël Cosnard

Table des matières

1	Préambule	8
1.1	Objet du document.....	8
1.2	Glossaire	8
2	Documents applicables.....	9
2.1	Exigences	9
2.2	Transverse	10
2.3	Projet.....	14
2.3.1	Exemple de document spécifique projet ou MOE.....	14
3	Contexte Technique SI	15
4	Socles Techniques	16
4.1	SX	16
4.1.1	Cycle de vie.....	17
4.1.2	Lien avec Kafka.....	17
4.2	Hawai	19
4.3	PFS	19
4.3.1	Particularités K8S UCN	20
4.4	PFSv2	22
4.5	Médiation événementielle - Kafka	23
4.6	Stockage Cloud : S3 & CEPH	23
4.6.1	Proxy Transfert.....	24
4.7	FullStack	24
4.7.1	FullStack REST Backend	24
4.7.2	FullStack SPA (Front).....	24
4.7.3	FullStack PFS HELM.....	24
4.7.4	FullStack Communication.....	25
4.8	ArchiMed	26
5	Principes de conception.....	27
5.1	Généralités.....	27
5.1.1	Langages et librairies	27
5.2	Patterns généraux.....	27
5.2.1	Idempotence.....	27
5.2.2	Couplage lâche	27
5.2.3	CQRS	28
5.2.4	Coupe circuit	28
5.2.5	Sûreté de fonctionnement	28
5.2.6	Impact.....	29
5.2.7	Supervision et Observabilité des traitements	29
5.2.8	Traçabilité des usages.....	29
5.2.9	Contract first.....	29
5.2.10	Verrous optimistes.....	29
5.3	API	29
5.3.1	Généralités	29
5.3.2	Archimed	30
5.3.3	Bouchons	31

5.3.4	Consommation adaptative.....	31
5.4	Batch.....	31
5.4.1	Framework	31
5.4.2	Planification et supervision de la planification	32
5.5	Traitement des messages.....	33
5.5.1	Cas d'usage	33
5.5.2	Kafka - Sécurisation et ACL	33
5.5.3	Nomenclature des topics.....	34
5.5.4	Partitionnement	34
5.5.5	Structure des messages.....	35
5.5.6	Consommation	35
5.5.7	Mise à l'échelle.....	35
5.5.8	Commit.....	36
5.5.9	Evolutions en cours	36
5.6	Stockage S3.....	36
5.7	Logs	36
5.7.1	Contenu.....	37
5.7.2	Emplacement	37
5.7.3	Niveaux de logs.....	37
5.7.4	Configuration.....	37
5.8	Base de données	38
5.8.1	Généralités	38
5.8.2	BDD SNV2 SX.....	39
5.8.3	BDD Nationales.....	39
5.9	IHM	40
5.9.1	Intégration PORA	40
5.9.2	Intégration BIPE	40
5.9.3	Consommation d'APIs	40
5.9.4	Stockage	41
5.9.5	Composants réutilisables	41
5.9.6	Format des images.....	41
5.9.7	Nettoyage des exemples fullstack.....	41
5.9.8	Bonnes pratiques	41
5.10	Conteneurisation	41
5.10.1	Construction des images.....	41
5.10.2	Plateformes cibles	41
5.10.3	Particularités sur PFS de dev et staging.....	46
5.11	Tests	46
5.12	Divers.....	46
5.12.1	Conventions de nommage.....	46
5.12.2	Particularités sur environnement de dev et incubateur.....	46
5.12.3	Commentaires et lisibilité du code.....	47
6	Supervision.....	48
6.1	Sondes.....	48
6.1.1	Principe général	48
6.1.2	Tests de vie personnalisés	48

6.2	Métriques	50
7	Sécurité	51
7.1.1	Sécurité des développements bonnes pratiques	51
7.1.2	Composants utilisés et vulnérabilités	51
7.1.3	Charte d'application.....	51
7.1.4	Contrôle des dépendances.....	51
8	Chaine de fabrication.....	52
8.1	Outils.....	52
8.2	Intégration continue.....	52
8.2.1	Git.....	52
8.2.2	Jenkins	53
8.2.3	Nexus	54
8.2.4	Sonarqube.....	54
8.2.5	Dependency Check	55
8.3	Livraison.....	56
8.3.1	Lot Refonte	56
8.3.2	Livraison SNV2.....	57
8.3.3	Livraison PFS	62
8.3.4	Livraison Hawai	64
8.4	Procédures à suivre	66
8.4.1	Construction d'application Frontend	66
8.4.2	Construction d'application Backend.....	66

1 Préambule

1.1 Objet du document

La Spécification Technique Générale (STG) vient en complément du DAT.

Le document décrit le cadre référentiel en termes de normes applicables et exigences techniques à prendre en compte dans la conception détaillée et l'implémentation. Il ne rentre pas dans le détail des spécifications par composant mais donne le point de départ des spécifications détaillée et l'orientation à leur donner.

Les concepts spécifiés de manière générale sont des exigences techniques qui doivent être prises en compte. La rédaction est organisée de manière synthétique visant à permettre une bonne compréhension du contexte et des contraintes. La STG est accompagnée d'un ensemble d'exigences applicables permettant une meilleure traçabilité de certains aspects traités.

● #modif

Les normes des langages utilisés sont publiques et doivent donc être respectées (conventions de nommage, structure du projet, commentaires et documentations utiles, nombre de paramètres d'une méthode ...etc.) sans qu'elles soient mentionnées explicitement dans ce document.

1.2 Glossaire

Terme	Définition
RAF	Recouvrement Amiable et Forcé
OBPA	Obligation De Payer
SNV2	Système National Version 2
CDC	Capture Data Change
KAFKA	Brique évènementielle permettant de capturer les mouvements de l'écart négatif.
ECNG	Objet métier du Legacy (Ecart négatif)
TA	Travail agent (<i>WATT / lien applicatif pour remonter des informations aux gestionnaires</i>)
BOA	Back Office Assistance
Créance	Objet métier du nouveau système (RAF Rénové)
CM2	Chaîne de traitement s'appuyant sur des composants techniques (Kafka Connect; KAFKA) et applicatif (ex. : traitement 1ECM) qui capture les mouvements sur un ensemble de données (dont les écarts négatifs) et les transforme en événements fonctionnels.
API	Acronyme d'Application Programming Interface, que l'on traduit en français par interface de programmation d'application.
CRUD	Acronyme informatique anglais CRUD (pour create, read, update, delete)
Protéine	Process d'architecture continue

2 Documents applicables

Les documents applicables au projet sont référencés ci-dessous. Ces normes et documentations prévalent en cas de doute sur la spécification d'un composant.

2.1 Exigences

● #modif

Nom	Exigences Non Fonctionnelles
Description	Ce document vient compléter la STG avec un ensemble d'exigences non fonctionnelles applicables au programme.
Référence	DAFA_ENF-3.5.0.pdf BL-DAIS-DAOS-ENFAPORT-310725-1106-14216.pdf BL-SSI-310725-1109-16002.pdf DSI-ENFAPORT-170226-1450-32278.pdf

2.2 Transverse

Les documents listés sont accessibles via les liens fournis dans les descriptions. Ils sont aussi fournis en Annexe de la STG.

ID de l'exigence	Libellé Exigence	Description
1290	API Backend - Recommandations techniques/SDED - BT - Normes Backend	Le développement et la conception des services backend exposés doivent suivre les principes décrits dans le document: SDED - BT - Normes Backend ----- https://readthedocs.cnp.recouv/docs/fullstack/fr/latest/normes/backend.html http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/Fullstack/fullstack.zip
1293	API Backend - Recommandations techniques/SDED - BT - Normes Backend - REST	Le développement et la conception des services des services REST exposés doivent suivre les principes décrits dans le readthedocs » Fullstack » Fullstack - Backend Rest & Kafka » Les normes » Norme Backend (ex document: SDED - BT - Normes Backend - v1.6.2.a.pdf) --- https://readthedocs.cnp.recouv/docs/fullstack/fr/latest/normes/rest/index.html http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/Fullstack/fullstack.zip
1291	API Backend - Recommandations techniques/SDED - BT - Normes Backend - REST - Conception et patterns rest	Le développement et la conception des services REST exposés doivent suivre les principes décrits dans le readthedocs référencé dans l'exigence : 1294
1294	API Backend - Recommandations techniques/SDED - BT - Normes et documentation	Le développement doit suivre les principes décrits dans le site readthedocs : https://readthedocs.cnp.recouv/docs/fullstack/fr/latest/normes/index.html > Norme : Journalisation et traces applicatives backend ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/Fullstack/fullstack.zip
977	Archimed v2 - Recommandations techniques et normes	La construction et le déploiement des bundles doit suivre les recommandations techniques indiquées dans : https://confluence.urssaf.recouv/display/AV2/Accueil+ARCHIMED+V2 ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/Archimed/Confluence_Archimed_v2.pdf
1901	Backend - SMTP	La documentation pour l'envoi d'emails en SMTP se trouve dans le document suivant ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Divers/Guide_des_bonnes_pratiques_SMTP.pdf
1297	Backend/SNV2 - Documentation Socle SX	Les règles et la documentation pour l'intégration d'applicatifs dans le socle SX se trouvent dans le readthedocs dédié ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/SX-SNV2/sx5-documentation.zip
1805	Backend/SNV2 - Editique et POSDOC	La documentation pour l'édition de documents est rassemblée dans le dossier suivant ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Divers/Documentation%20POSDOC
1296	Backend/SNV2 - Recommandations techniques/SDED - BT - Container Webapp java régionale - v2.2.pdf	La définition de éléments de construction d'un conteneur sur socle SXs doivent suivre les principes décrits dans les documents: SDED - BT - Container Webapp java régionale - v2.2.pdf ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/SX-SNV2/SDED%20-%20BT%20-%20Container%20Webapp%20java%20r%C3%A9gionale%20-%20v2.2.pdf
1305	Base de données - Recommandations techniques/Oracle- PostgreSQL/ACOSS-DAT-SX-PG-V1.0	La description des solutions mises en œuvre pour la migration oracle vers postgresql est décrite dans le document: ACOSS-DAT-SX-PG-V1.0.pdf ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/SX-SNV2/ACOSS-DAT-SX-PG-V1.0.pdf
1298	Batch - Recommandations techniques/SDED - BT - TECH - Normes Batch java régional	Le développement des batchs doit suivre les normes décrites dans le document: SDED - BT - TECH - Normes Batch java régional - v1.3.pdf ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/SX-SNV2/SDED%20-%20BT%20-%20TECH%20-%20Normes%20Batch%20java%20r%C3%A9gionale%20-%20v1.3.pdf

ID de l'exigence	Libellé Exigence	Description
1299	Batch/SNV2 - Recommandations techniques/nomenclature_code_traitement_snv2	La nomenclature des code de traitement doit suivre les normes décrites dans le readthedocs sx5 https://readthedocs.cnp.recouv/docs/sx5-documentation/fr/latest/SNV2/autresdocuments/nomenclaturetraitement.html?highlight=nomenclature exporté dans sx5-documentation.zip ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/SX-SNV2/sx5-documentation.zip
1300	Batch/SNV2 - Recommandations techniques/systeme_de_commandes_cmde	La nomenclature des code de traitement doit suivre les normes décrites dans le document: "systeme_de_commandes_cmde_.doc" ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/SX-SNV2/systeme_de_commandes_cmde_.doc
1307	Bus de messages - Recommandations techniques/SDED - kafka - accrochage_plateforme_kafka	Les normes de communication à appliquer avec la plateforme événementielle sont décrits dans le document : SDED - kafka - accrochage_plateforme_kafka v0.1.pdf ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/Kafka/SDED%20-%20kafka%20-%20accrochage_plateforme_kafka%20v0.1.pdf
1308	Bus de messages - Recommandations techniques/SDED - kafka - gestion_des_transactions	Les normes de communication à appliquer avec la plateforme événementielle sont décrits dans le document: SDED - kafka - gestion_des_transactions v0.1.pdf ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/Kafka/SDED%20-%20kafka%20-%20gestion_des_transactions%20v0.1.pdf
1306	Bus de messages - Recommandations techniques/SDED - Normes Kafka	Les normes de communication à appliquer avec la plateforme événementielle sont décrits dans le site read the doc: Kafka ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/Kafka/documentation-kafka.zip
978	Chaîne de construction - Recommandations techniques/GuidesTechnique	La construction et le déploiement d'artefacts doit suivre les recommandations techniques indiquées dans : https://confluence.urssaf.recouv/display/UL/Usine+Logicielle ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Industrialisation/UsineLogicielle.pdf
979	Chart Helm - Recommandations techniques	Les bonnes pratiques décrites dans le chapitre « Bonnes Pratiques, Cas d'usages et Normes » du tutoriel « Tutoriel MOE » présent dans le read the docs Documentation PFS doivent être suivies lors de la création du chart helm dédié à la PFS. ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/PFS/documentation-pfs.zip
971	Compte technique - Déclaration des droits Anais	Le fabricant doit demander la déclaration des droits métiers Anais. Le modèle de déclaration de droits métiers a utiliser est Anais_formulaire-appli_XXX V4.5.xlsx ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Authentication/Anais_formulaire-appli_XXX%20V4.8.xlsx
1322	COTS - Recommandations techniques/SDED - BT - TECH - Gestion des versions des dépendances et des composants	Le développement et la définition de l'application modèle et de ses principes de développement doit suivre les principes décrits dans le readthedocs https://readthedocs.cnp.recouv/docs/fullstack/fr/latest/normes/normelibraries.html?highlight=composants ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/Fullstack/fullstack.zip
990	Documentation/Cloud - Documentation et support/Ceph-S3	La documentation applicable pour l'utilisation des infrastructure de stockage Ceph/S3 se trouve ici : https://confluence.urssaf.recouv/pages/viewpage.action?pageId=167412084 http://exploit-wikijs.cnp.recouv/fr/documentations/transfert-de-fichiers/Vue-densemble ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/Cloud/Normes%20et%20C3%A9ferentiel%20Ceph&S3%20-%20201.5.pdf
985	Documentation/Hawai - Formation	L'hébergement des Applications Web et des Applications Intranet est présentée dans le document: http://techniques.renov.recouv/doku.php?id=hawai:externe:formations:start
983	Documentation/Hawai - hawai_v6	L'hébergement des Applications Web et des Applications Intranet est présentée dans la page confluence dédiée au Socle HAWAI 6.5 : http://confluence.urssaf.recouv/display/HW/Socle+HAWAI+6.5 . et dans la page spécifique à la gestion des bases de données https://readthedocs.cnp.recouv/docs/fullstack/fr/latest/fullstack/fullstack-rest-backend/plusloin/bdd_hawai.html . ----- https://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/blob/3.5.0/Documents%20Techniques%20Applicables/Socles/Hawai/Confluence_Hawai_6.5.pdf

ID de l'exigence	Libellé Exigence	Description
986	Documentation/Hawai - Intégration	L'hébergement des Applications Web et des Applications Intranet est présentée dans la page : https://readthedocs.cnp.recouv/docs/fullstack/fr/latest/fiches/hawai.html?highlight=hawai ainsi que le document guide_d_integration_hawai-v1.9.3.doc ----- https://readthedocs.cnp.recouv/docs/fullstack/fr/latest/normes/rest/index.html http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/controle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/Fullstack/fullstack.zip http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/controle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/Hawai/guide_d_integration_hawai-v1.9.3.doc
984	Documentation/Hawai - Présentation générale	L'hébergement des Applications Web et des Applications Intranet est présentée dans le document : http://techniques.renov.recouv/doku.php?id=hawai:externe:hawai6:start
988	Documentation/PFS - Documentation	Décrit les bases du déploiement de charges de travail sur la PlateForme de Service : documentation-pfs.zip ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/controle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/PFS/documentation-pfs.zip
989	Documentation/PFS - Documentation et support	Décrit les bases du déploiement de charges de travail sur la PlateForme de Service ----- https://gitlab.cnp.recouv/docs/support-et-documentation
987	Documentation/PFS - Tutoriel PFS	Décrit les bases du déploiement de charges de travail sur la PlateForme de Service : Read The Doc PFS > Tutoriel MOE ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/controle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/PFS/documentation-pfs.zip
982	Documentation/SNV2 - Saturne - Présentation Générale.pdf	Le fonctionnement du Socle technique SX historique version 5 est présenté dans le document: Saturne - Présentation Générale.pdf ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/controle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/SX-SNV2/Saturne%20-%20Pr%C3%A9sentation%20G%C3%A9n%C3%A9rale.pdf
980	Documentation/SNV2 - sx5- documentation	Le fonctionnement du Socle techniqueSX historique version 5 est présenté dans le document: sx5- documentation.zip ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/controle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/SX-SNV2/sx5-documentation.zip
981	Documentation/SNV2 - Sx5Web	Le fonctionnement du Socle technique SX historique version 5 est présenté dans le document: Sx5Web.pdf ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/controle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/SX-SNV2/Sx5Web.pdf
1320	Dépot - Recommandations techniques/Chaine_FabV2_Java	La structuration Référentiels Git et Nexus à suivre sont décrites dans le document Chaine_FabV2_Java_V1.1.pdf " ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/controle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/SX-SNV2/Chaine_FabV2_Java_V1.1.pdf
1303	Frontend - Charte graphique/Charte Ergonomique Acooss	La charte graphique et les règles d'ergonomie applicables sont décrites sur le site https://kxzkx5.axshare.com/?id=clwcqx&p=une_charte_ergonomique__pourquoi__ ----- https://kxzkx5.axshare.com/?id=clwcqx&p=une_charte_ergonomique__pourquoi__
1323	Frontend - Norme accessibilité ISO	Les IHM développées doivent respecter la norme ISO 4050 ----- https://www.w3.org/WAI/standards-guidelines/wcag/fr
1302	Frontend - Recommandations techniques/SDED - BT - Norme Frontend	Le développement et la conception d'élément du front end Angular doivent suivre les principes décrits dans le readthedoc Docs » Les normes applicatives » Norme frontend (ex document: SDED - BT - Norme Frontend v1.1.0.pdf) ----- https://readthedocs.cnp.recouv/docs/fullstack/fr/latest/normes/frontend.html http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/controle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/Fullstack/fullstack.zip
1301	Frontend - Recommandations techniques/SDED - BT - Normes Frontend - Communication SPA	Le développement et la conception d'élément du front end Angular doivent suivre les principes d'utilisation de bibliothèque de communication décrits dans le readthedoc Docs » Les normes applicatives » Norme d'architecture Communication IHM Web ----- https://readthedocs.cnp.recouv/docs/fullstack/fr/latest/normes/comihm.html http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/controle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/Fullstack/fullstack.zip

ID de l'exigence	Libellé Exigence	Description
1324	Frontend - Utilisation de BIPE	Les IHM développées doivent respecter les recommandations décrites dans la documentation de BIPE (Brique des IHM de Pilotage des Ecrans) ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Ergonomie/documentation_bipe.zip
1064	Livraison/SNV2 - Guichet livraison documentaire/GLD_Guide_utilisateur	La procédure de dépôt documentaire à suivre est décrit dans le document: GLD_Guide_utilisateur.pdf ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.3.0/Documents%20Techniques%20Applicables/Socles/SX-SNV2/GLD_Guide_utilisateur.pdf
1312	Log - Normes Qelar	Les applications déployées en production dans le cadre du SI de l'URSSAF CN doivent couvrir les exigences de la norme QELAR (Qualité d'Exploitation des Livrables des Applications du Recouvrement) décrites dans le document "Norme QELAR V2.0.docx" et le readthedoc https://readthedocs.cnp.recouv/docs/norme-qelar/fr/latest/ exporté en zip. ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Supervision/Norme%20QELAR%20V2.0.docx http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Supervision/norme-qelar.zip
1321	Modèle de branchement - Recommandations techniques/SDED - BT - FullStack - Workflow de développement	La gestion des branches et des versions doit suivre les préconisations faites dans le document: SDED - BT - FullStack - Workflow de développement v1.2.0.b.pdf ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/M%C3%A9thode/SDED%20-%20BT%20-%20FullStack%20-%20Workflow%20de%20d%C3%A9veloppement%20v1.2.0.b.pdf
1310	Observabilité - Recommandations techniques/CLEA_PRJ2_DSN_STD_Traces_Empreintes	Les règles de productions de traces probantes sont décrites dans le document: "CLEA_PRJ2_DSN_STD_Traces_Empreintes.pdf" ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Supervision/CLEA_PRJ2_DSN_STD_Traces_Empreintes.pdf
1311	Observabilité - Recommandations techniques/Empreinte_Traces_ModeOperatoire - Pivot	Les règles de productions de traces probantes sont décrites dans les documents : • Empreinte_Traces_ModeOperatoire - Pivot V.pdf • Empreinte_Traçabilité_des_usages_métiers_sensibles.pdf ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Supervision/Empreinte_Traces_ModeOperatoire%20-%20Pivot%20V.pdf http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Supervision/Empreinte_Traçabilité_des_usages_métiers_sensibles.pdf
991	PFS - Documentation de référence	Les principes de mise en œuvre d'une application sur la PFS sont décrits dans le readthedoc : https://readthedocs.cnp.recouv/docs/documentation-pfs/fr/latest/ressources/index.html ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/PFS/documentation-pfs.zip
992	PFS - Exemple et documentation	Des supports complémentaire de documentation sur la PFS sont stocké dans le dépôt GIT : https://gitlab.cnp.recouv/docs/support-et-documentation
3655	PFSv2 - Documentation	Décrit les bases du déploiement de charges de travail sur la PlateForme de Service : https://confluence.urssaf.recouv/display/PFSACC/PFSv2+-+Documentation+utilisateurs ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/PFSv2/Confluence_PFS_v2.pdf http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/Confluence_PFS_v2_Externalisation_des_logs.pdf
1318	Qualité - Recommandation technique/SDED - BT - Norme qualité - SonarQube	Les sources développées doivent répondre aux exigences qualité définies dans le readthedoc fullstack Docs » Les normes applicatives » Norme Qualité (ex document "SDED - BT - Norme qualité - SonarQube") ----- https://readthedocs.cnp.recouv/docs/fullstack/fr/latest/normes/qualite.html http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/Fullstack/fullstack.zip
1313	Sécurité - Recommandations techniques/Corpus documentaire Sécurité	Les bonnes pratiques de sécurité et de gestion de la protection des données à suivre sont décrites dans les documents contenus dans l'archive "Corpus documentaire Sécurité" ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/S%C3%A9curit%C3%A9/Corpus%20documentaire%20S%C3%A9curit%C3%A9.zip

ID de l'exigence	Libellé Exigence	Description
1316	Sécurité - Recommandations techniques/SDED-BT - memo - bonnes pratiques ouverture code source	Les bonnes pratiques de sécurité et de gestion de la protection des données à suivre sont décrites dans le document: SDED-BT - memo - bonnes pratiques ouverture code source_060219_v0.2.pdf ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/M%C3%A9thode/SDED-BT%20-%20memo%20-%20bonnes%20pratiques%20ouverture%20code%20source_060219_v0.2.pdf
1317	Sécurité - Recommandations techniques/SDED-BT-RGPD-SDLC	Les bonnes pratiques de sécurité et de gestion de la protection des données à suivre sont décrites dans les documents contenus dans le document SDED-BT-RGPD-SDLC_v1.1.pdf ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/S%C3%A9curité%C3%A9/SDED-BT-RGPD-SDLC_v1.1.pdf
1314	Sécurité - Recommandations techniques/SDED-BT_exigences sécurité codage	Les bonnes pratiques de sécurité et de gestion de la protection des données à suivre sont décrites dans le document: SDED-BT_exigences sécurité codage_v1.0.1.pdf ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/S%C3%A9curité%C3%A9/SDED-BT_exigences%20s%C3%A9curité%C3%A9%20codage_v1.0.1.pdf
1315	Sécurité - Recommandations techniques/SDED-BT_standard sécurité codage SPA	Les bonnes pratiques de sécurité et de gestion de la protection des données à suivre sont décrites dans le document: SDED-BT_standard sécurité codage SPA_v1.0.pdf ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/S%C3%A9curité%C3%A9/SDED-BT_standard%20s%C3%A9curité%C3%A9%20codage%20SPA_v1.0.pdf
1418	Sécurité/PFS - Règles de sécurité Cilium	Le document décrivant les règles de sécurité cilium à appliquer sont décrites dans le document: https://recouv.sharepoint.com/sites/PlateformeDigitale/_layouts/15/Doc.aspx?sourcedoc=%7B2C06F73D-D71C-4B08-AB1F-57C21587AC17&file=CiliumNetworkPolicies_v1.0(en%20validation).docx&action=default&mobileredirect=true&DefaultItemOpen=1
1325	Tests - Automatisation des tests	La mise en place des tests doit suivre les préconisations faites dans les documents: • Norme automatisation des tests (readthedocs https://readthedocs.cnp.recouv/docs/fullstack/fr/latest/normes/testsauto/index.html) ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Socles/Fullstack/fullstack.zip
1319	Tests - Recommandations techniques	La mise en place des tests doit suivre les préconisations faites dans les documents: • ACOSS-Guide de pratique des tests MOE • Guide_operationnel_GestionTests • STE_Politique de Test ----- http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Tests/STE_Politique%20de%20Test_V2.2.pdf http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Tests/URSSAF%20CN-Guide%20de%20pratique%20des%20tests%20MOE.v1.7.pdf http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/contrôle-et-recouvrement-amiable-force-parametres/raf/raf-referentiel-projet/raf-document-techniques-applicables/-/raw/3.5.0/Documents%20Techniques%20Applicables/Tests/Guide_operationnel_GestionTests%20-%20V2.0.pdf

2.3 Projet

2.3.1 Exemple de document spécifique projet ou MOE

Nom	STD Projet
Description	Spécification Techniques liées au projet
Référence	Lien GitLab http://gitlab.altair.recouv/sded/coeur-de-metier-r.....

- #modif #3.2

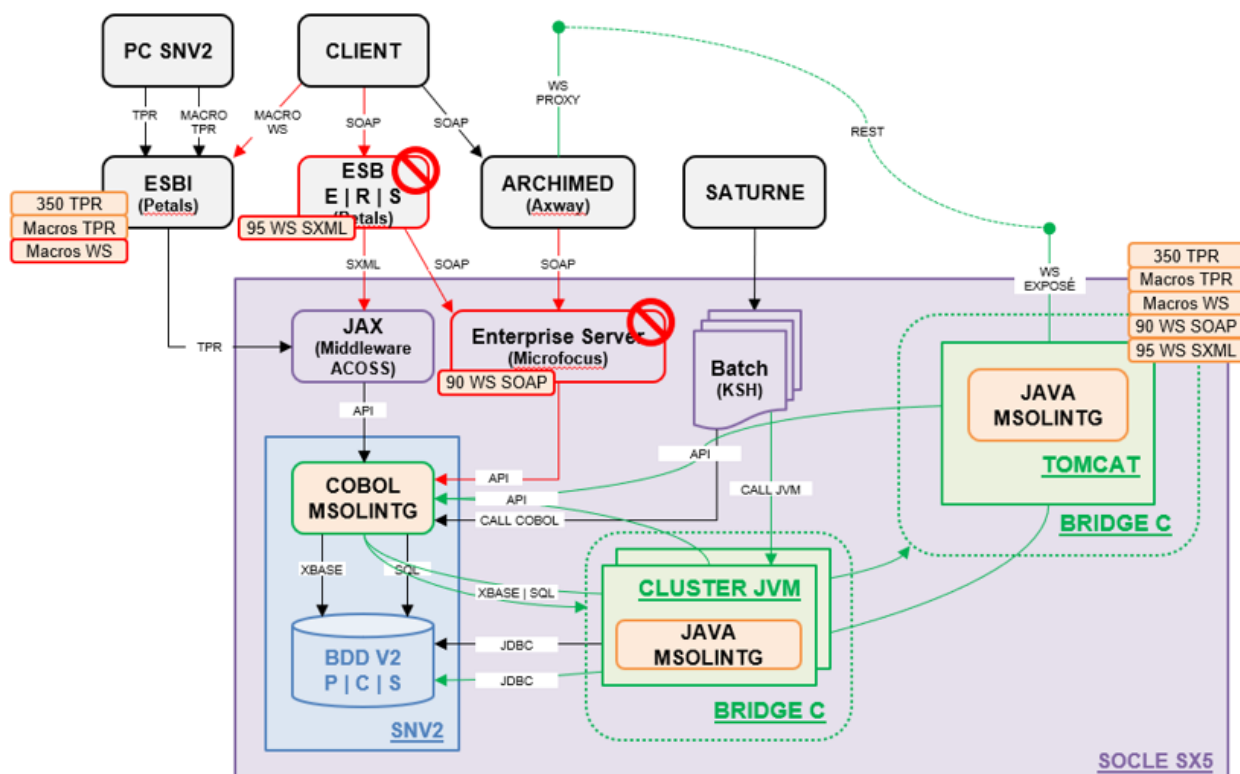


4 Socles Techniques

Sous l'appellation « socles techniques » sont regroupés différents types de composants logiciels utilisés dans le SI URSSAF. Il y a des plateformes de déploiement, des systèmes de médiation et des frameworks applicatifs. Ils permettent de normaliser les applications développées et leur exploitation et ils fournissent des outils, procédures et normes pour accompagner leur implémentation.

La solution développée doit se baser sur ces différents socles décrits ci-dessous et suivre les préconisations techniques qui leur sont associés. Dans la dernière version disponible.

4.1 SX



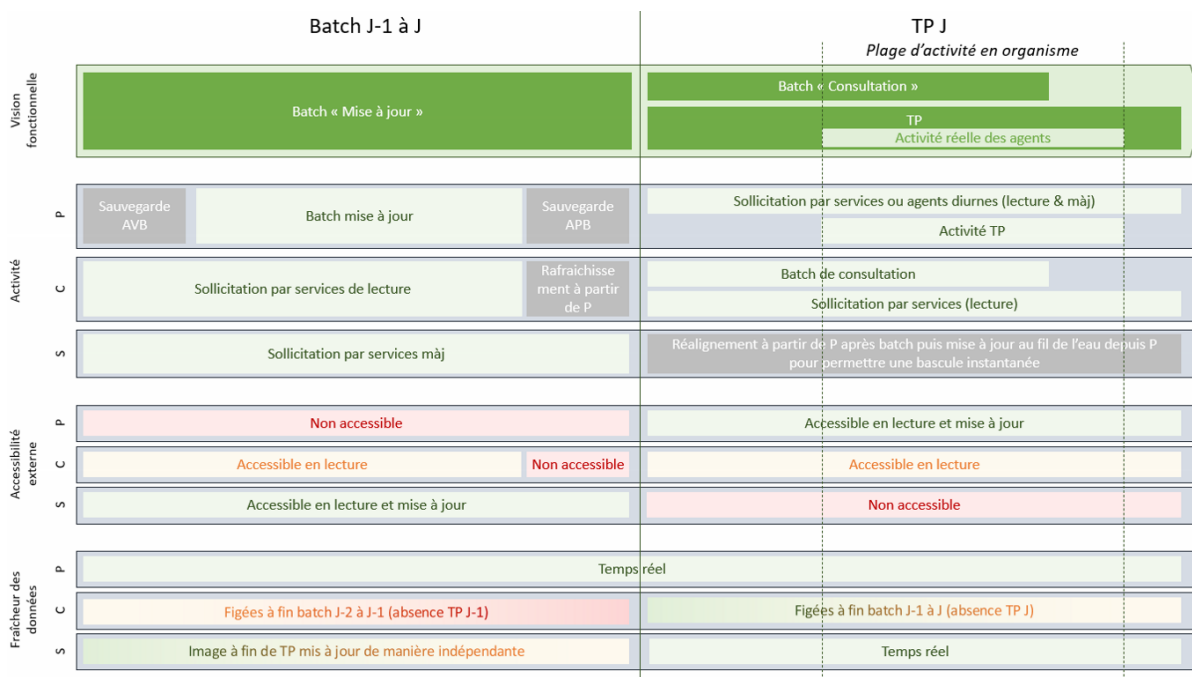
● #modif #3.2

- Une plateforme SX héberge le cœur de métier Informatique d'un organisme (Urssaf régional ou spécifique) appelé SNV2 (Système National V2)
- Toutes les plateformes SX5 sont identiques : elles possèdent un socle technique et un applicatif commun : le SNV2.
- En production, il existe 39 bases SNV2 sur leurs plateformes associées
- En production, il existe deux plages horaires de fonctionnement des serveurs SX :
 - La plage Transactionnel : permet l'accès à la base SNV2 en Interactif par le Transactionnel (PCSNV2), les services Web SXML, Microfocus, Movesol et Containers
 - La session Batch est planifiée par chaque Urssaf via le système de commandes. Elle est exécutée après fermeture des Urssafs.
 - Ces deux plages ne peuvent pas fonctionner en même temps.
- Le déploiement des services sur la plateforme SX se fait via la construction d'un lot applicatif SNV2
- Les spécificités des différentes plateformes déployées sont paramétrées via des variables « castor » accessibles par les programmes

Les composant applicatifs régionaux de la solution doivent s'intégrer dans le plan de production SNV2 actuel en accord avec les recommandations et normes correspondantes. Ces derniers devront être exploitables via le « Portail Technique » et comporter le paramétrage (Castor) associé.

4.1.1 Cycle de vie

Pour chaque région il existe plusieurs serveurs et base de données qui sont accessible en fonction du cycle de vie du SNV2 ce dernier comporte un plage horaire batch et une plage transactionnel.



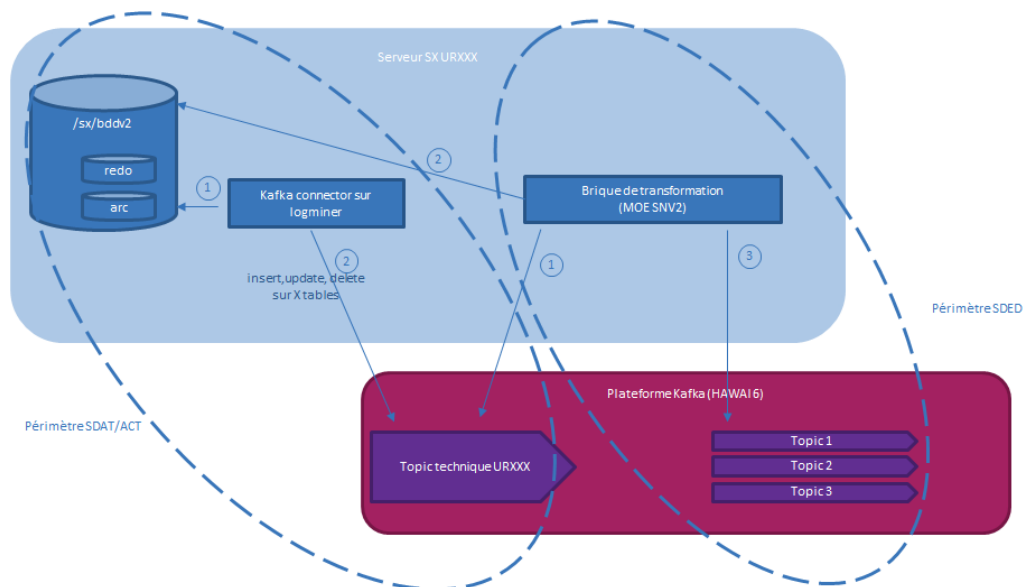
4.1.2 Lien avec Kafka

4.1.2.1 Change Data Capture

Un mécanisme de « change data capture » basé sur debezium 1.7.2 est en place sur la base de données et doit être utilisé pour identifier les évolutions de différentes entités métier.

Le CDC n'est pas actif sur toute la journée il est interrompu durant les plages horaires des batch de nuit afin qu'il ne soit pas affecté par une restauration de base de données. Il dépile l'ensemble des modifications au redémarrage du transactionnel le matin.

Une table d'évènements technique dans la base de données permet de faire remonter ces derniers dans le CDC.



En fonctionnement sur une base de données PostgreSQL Debezium utilise les informations de pgoutput.

Des informations détaillées sur le fonctionnement de Debezium et le format des messages peuvent être récupéré sur le site officiel : <https://debezium.io/documentation/reference/1.7/>

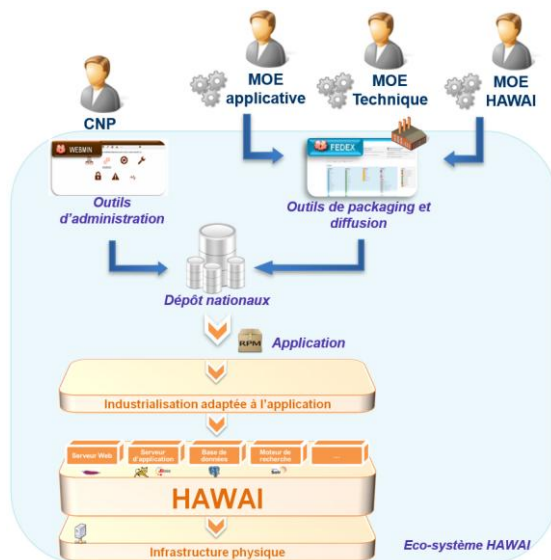
Exemple de messages kafka pour une mise à jour de données :

```
{
  "schema": { ... },
  "payload": {
    "before": {
      "ID": 1004,
      "FIRST_NAME": "Anne",
      "LAST_NAME": "Kretchmar",
      "EMAIL": "annek@noanswer.org"
    },
    "after": {
      "ID": 1004,
      "FIRST_NAME": "Anne",
      "LAST_NAME": "Kretchmar",
      "EMAIL": "anne@example.com"
    },
    "source": {
      "version": "1.6.0.Final",
      "name": "server1",
      "ts_ms": 1520085811000,
      "txid": "6.9.809",
      "scn": "2125544",
      "commit_scn": "2125544",
      "snapshot": false
    },
    "op": "u",
    "ts_ms": 1532592713485
  }
}
```

4.1.2.2 Publication du cycle de vie

Le socle SX publie des messages dans Kafka pour notifier les passage en mode batch ou transactionnel.
[Notifications evenementielle : start/stop TP — Documentation SX5, SNV2 et FABV2 1.0.0 \(cnp.recouv\)](#)

4.2 Hawai

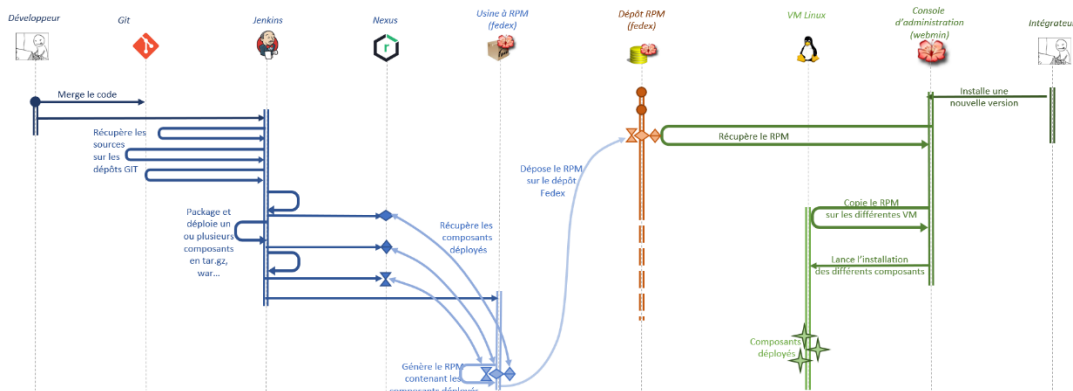


Le socle Hawai fourni un ensemble d'outils pour le packaging, le provisionning, le déploiement et l'administration des applications. Le socle Hawai met à disposition de nombreux services de base pour l'exécution de l'application (Tomcat, jBoss, apache, postgresql, solr, activemq, bpm...).

La mise en place d'une application sur ce socle suit un cycle de vie et une comitologie spécifique.

Les application hawai doivent être configurables via des fichiers comportant des variables injectées par l'environnement.

Le livrable à destination des plateformes HAWAI doit être un RPM généré via l'outil FEDEX.



Une documentation détaillée de ce socle est disponible en interne via les liens suivant :

- [Produit HAWAI - Socle HAWAI - Confluence DSI Acoess \(urssaf.recouv\)](#) #modif #3.2
- <http://techniques.renov.recouv/doku.php?id=hawai:externe:hawai6:start>
- [L'intégration HAWAI \(Mode web\)](#)

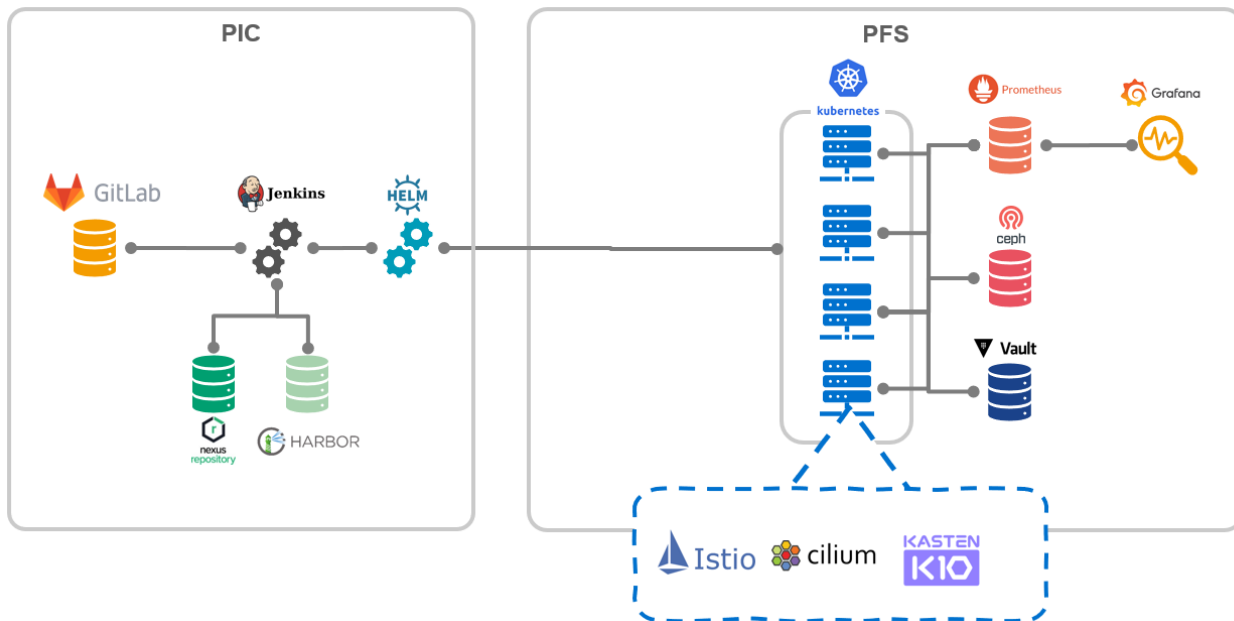
Certains composants applicatifs nationaux ou transverses sont déployés sur ce socle (Kafka)

Les serveurs de base de données pour les applications PFS doivent être déployés sur ce socle

4.3 PFS

La PFS est une plateforme interne de déploiement de charge de travail conteneurisées basée sur Kubernetes. Elle est utilisée pour les déployer de applicatifs nationaux.

Elle est composée des services suivants :



● #modif #3.2

Il s'agit d'une stack kubernetes standard avec quelques operateurs et des subchart helm obligatoires. Les subcharts obligatoires sont utilisés pour:

- La création des namespaces
- La création des buckets S3

Le socle PFS v1 va être décommissionné au profit de la PFSv2. Il ne doit plus être en cible pour les nouveau applicatifs.

4.3.1 Particularités K8S UCN

● #modif

4.3.1.1 Capsule

La création de namespaces et dépôts harbor nécessite la création d'un quota Capsule pour l'application. Cette création est réalisée par l'UCN à travers un ticket redmine aux équipes PFS.

4.3.1.2 CronJob

Les cronjob ne peuvent pas être utilisés pour la planification.

La gestion des cronjob est déléguée à l'outil automator et donc nécessite une configuration particulière. ([SDOPPE / CNP Job PFS · GitLab](#))

Dans le chart helm on utilise une ConfigMap pour stocker ce qui devrait être la config du job. Automator va lire à l'aide du script CJP la config et lancer ainsi un job synchrone.

4.3.1.3 Opérateurs

Les opérateurs suivants sont disponibles sur les k8s de la PFS

- Cilium
- Istio
- Kasten
- ArgoCD

4.3.1.4 Subcharts

Des subcharts sont mis à disposition pour traiter les cas suivants :

- Création de bucket S3
- Création des namespaces

4.3.1.5 Outillage

- Prometheus / Grafana pour observer les metriques standard ou custom des pod
- Opensearch pour accéder aux logs des pod d'un namespace

4.3.1.6 Deploy

● #modif

Le déploiement passe par un dépôt git « Deploy ».

Il comporte une branche master qui sert de template pour la valorisation de base du value.yaml et namespace.yaml. Le value.yaml sur cette branche n'est pas rempli. La branche master doit être livrée en même temps que les nouvelles versions du chart et elle doit être taguée en conséquence.

Le dépôt comporte ensuite une branche par environnement selon la nomenclature suivante :

- DEV/01
- DEV/02
- INT/01
- INT/07

La structure de ce dépôt est définie ici : https://gitlab.altair.recouv/sddos/innovation-et-transformation-digitale/procedure_deploiement_pfs/depotReference/-/tree/master

4.3.1.7 Bases De Données

● #modif

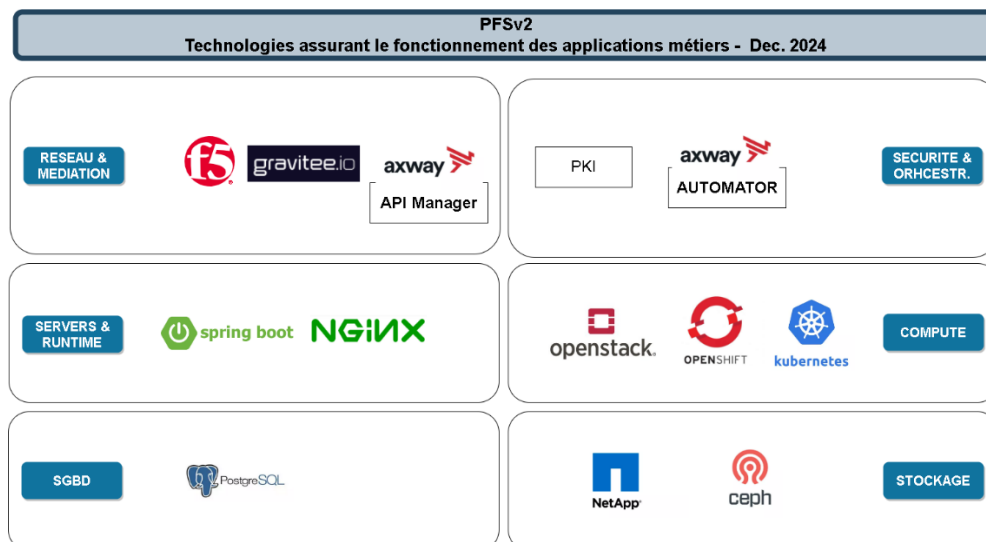
Les bases de données à destination de la PFS doivent être hébergées sur les environnements Hawaï. Elles doivent être mutualisés en créant plusieurs schémas sur un unique cluster de base de données pour la MOE.

4.4 PFSv2

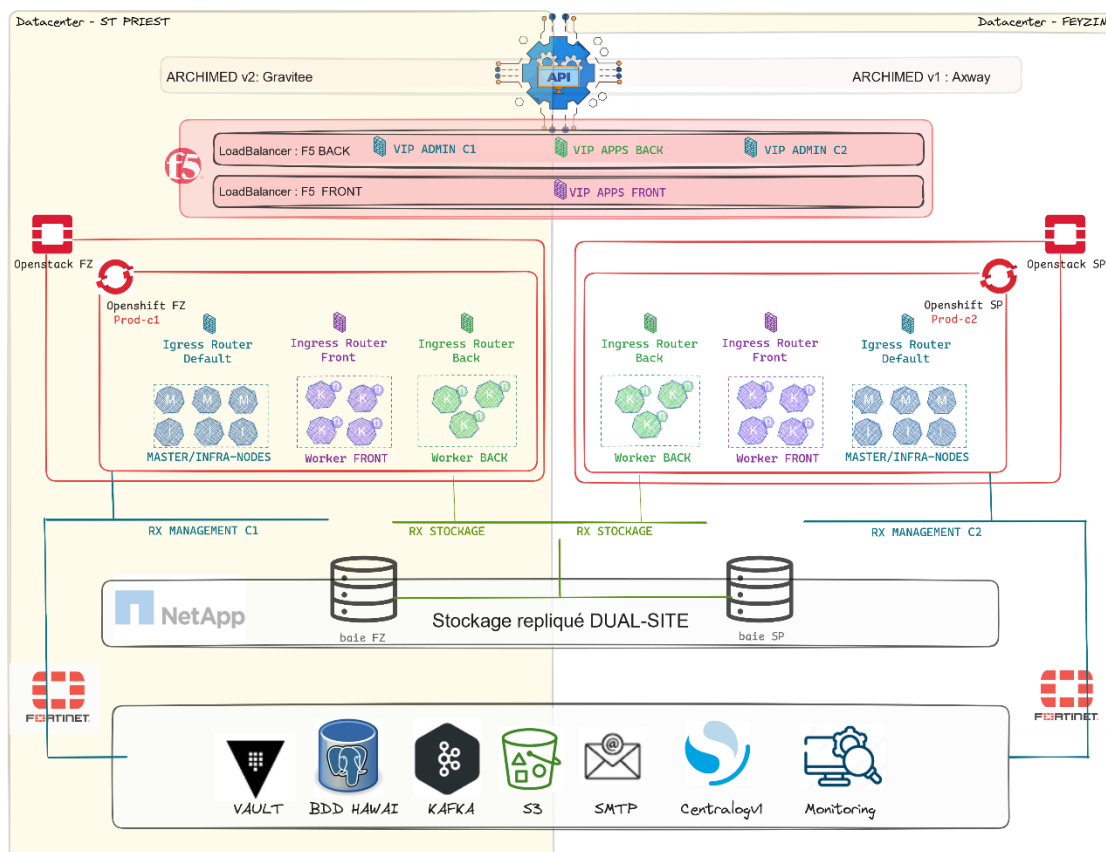
● #modif #3.4

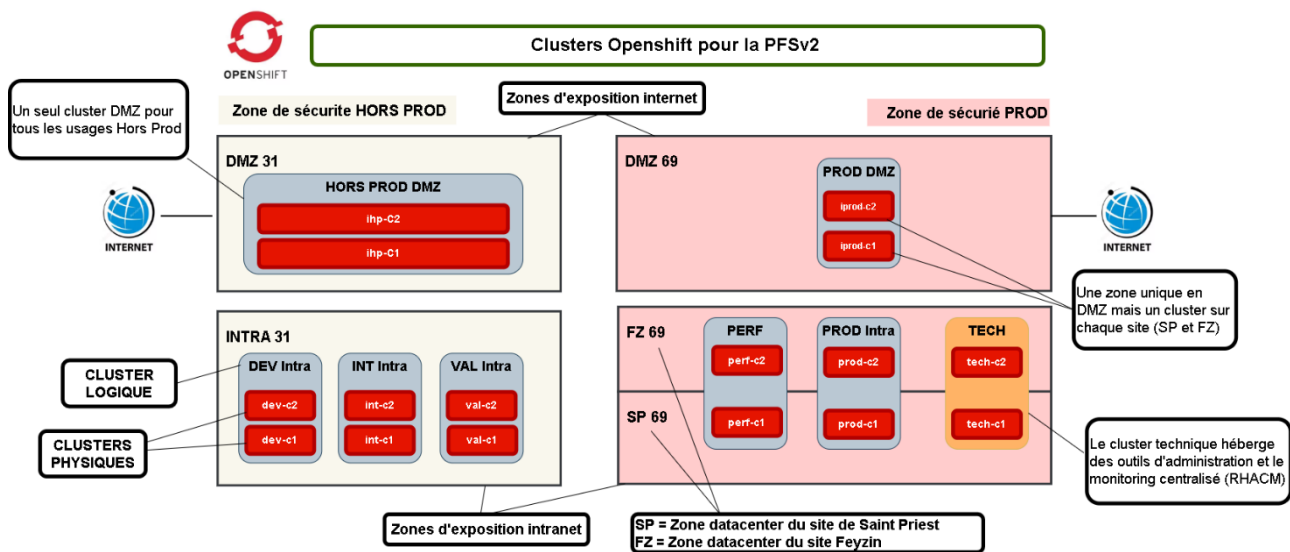
Le socle PFSv2 est basé sur la solution OpenShift.

Le socle PFSv2 est recommandé pour les composants applicatifs nationaux ou transverses.



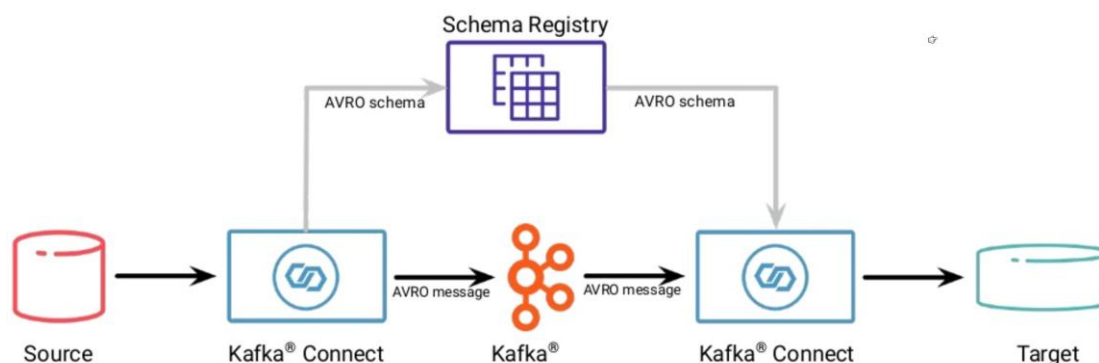
Le déploiement est fait en mode bi cluster





4.5 Médiation événementielle - Kafka

La médiation événementielle est basée sur kafka et son registry.



Les schémas sont gérés en formalisme **avro**. L'utilisation du registre est bientôt soumise à une norme en cours d'élaboration.

● #modif

L'accès aux topics est sécurisé via des politiques. Celles-ci sont déclarée via les bundles ArchiMed.

Il est donc nécessaire de se connecter à Kafka en oauth et de déclarer dans le bundle associé à l'appli les topic qu'elle gère. Dans le cas d'utilisation de topics gérés par d'autre applications il faut demander la mise à jour de leur bundle pour ouvrir les droits.

4.6 Stockage Cloud : S3 & CEPH

● #modif L'infrastructure CEPH (solution openSource de stockage distribuée) déployé à UCN propose trois protocoles :

- Bloc
- Fichiers
- Ceph Objet Storage (S3)

Le stockage objet S3 doit être utilisé dans le cadre des transferts de fichiers depuis ou vers la PFS.

4.6.1 Proxy Transfert

Le ProxyTransfert est un outil géré par les exploitants pour automatiser le transfert des fichiers déposés sur un bucket vers un serveur applicatif (hawai ou autre).

Dans le cas d'un fichier produit sur un bucket s3, l'état du transfert va être mis à jour dans le tag **pxtf** de l'objet en cours de transfert :

- consommé : le fichier est traité correctement jusqu'à sa destination.
- pris en compte : le fichier est en cours d'acheminement
- erreur : l'acheminement ne s'est pas passé correctement.

La configuration du transfert doit être décrite dans les documents d'exploitation et référencée dans la doc de lot.

4.7 FullStack

Les projets fullstack sont accessibles via les dépôts git suivants : <https://gitlab.altair.recouv/sdat/act/saul/fullstack>

● #modif

Ces projets doivent servir de point de départ et de référence en termes d'implémentation technique. Toutefois il s'agit de kit de démarrage démontrant l'implémentation de technologies pas toujours utiles au projet. Il convient donc de nettoyer les projets initialisés à partir de Fullstack.

4.7.1 FullStack REST Backend

La solution est basée sur l'application blanche « Fullstack Rest backend » fournie et maintenue par les équipes interne du Bureau Technique.

Cette application pose les bases de la structure projet et des librairies et frameworks à utiliser :

- Java
- Maven
- Spring Boot
- Hibernate

Gitlab : <https://gitlab.altair.recouv/sdat/act/saul/fullstack/backend/fullstack-backend> ● #modif

4.7.1.1 Compatibilité Java

Les développements ne devront pas utiliser de fonctions spécifiques aux implémentations des éditeurs afin de pouvoir être réutilisés sans contraintes dans d'autres implémentations.

Les développements JAVA doivent être compatibles à la fois sur OracleJDK et OpenJDK.

4.7.2 FullStack SPA (Front)

Les IHM de la solution seront basées sur l'application blanche « Fullstack SPA » fournie et maintenue par les équipes interne du Bureau Technique.

Cette application pose les bases de la structure projet et des librairies et frameworks à utiliser pour une IHM :

- Typescript
- Angular
- Bootstrap
- AgGrid

Elle contient aussi les modèles pour tous les composants d'IHM (formulaire, tables etc ...) selon la charte graphique attendue.

Gitlab : <https://gitlab.altair.recouv/sdat/act/saul/fullstack/frontend/fullstack-spa-intranet>

● #modif #3.2

4.7.3 FullStack PFS HELM

Contient la structure de base d'un chart HELM pour déployer une application en PFS.

Gitlab : <https://gitlab.altair.recouv/sdat/act/saul/fullstack/cloud/charts>

● #modif #3.2

4.7.4 FullStack Communication

Librairie de gestion des communication inter IHM.

Gitlab :

<https://gitlab.altair.recouv/sdat/act/saul/fullstack/frontend/communication-spa-angular> ● #modif

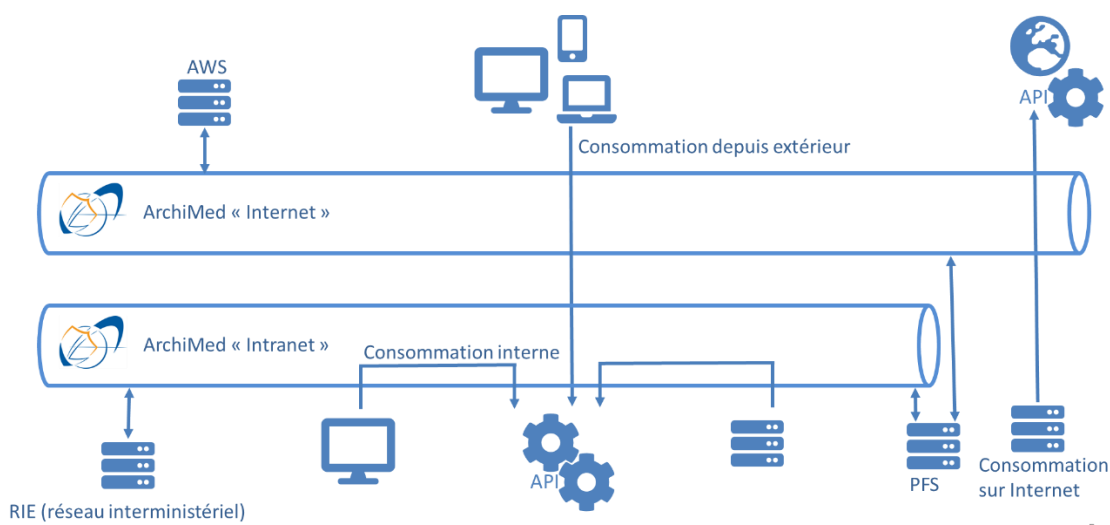
<https://gitlab.altair.recouv/sdat/act/saul/fullstack/frontend/communication-spa-core> ● #modif

4.8 ArchiMed

ArchiMed est l'écosystème qui gère : ● #modif #3.2

- Des **API gateways** (Gravitee) qui permettent de proxifier/sécuriser les services.
- Une brique sécurité : **PSS** qui va notamment gérer les jetons d'accès :
 - Jeton d'identité (id_token)
 - Jeton d'accès (access_token ou AT)
 - Jeton "historique" OCEAN
- Une brique permettant de gérer les accès provenant de l'extérieur du SI : **OIDCP**
- Un portail API

La plateforme proxyfie les services SOAP existant mais les nouveaux services sont en REST + OAuth



Les API créées pour la solution seront déployées via la plateforme ArchiMed. Et suivre la norme des contrats de services associée (§ Documents applicables).

5 Principes de conception

5.1 Généralités

5.1.1 Langages et librairies

● #modif #3.5

Les technologies retenues pour les applicatifs rénovés sont les suivantes :

Composant	Techno	Version
Langage de programmation Frontend	<i>Typescript</i>	5.8+
Langage de programmation Backend	<i>Java</i>	21
Framework Backend	<i>Spring Boot</i>	3.5.9
ORM	<i>Hibernate</i>	6.5+
Framework Frontend	<i>Angular,</i>	19+
SGBD	<i>PostgreSQL</i>	
Message brooker	<i>Kafka</i>	
Conteneurisation	<i>Kubernetes</i>	1.34+
Gestion de paquets k8s	<i>Helm</i>	3.16+
Interfaces services	<i>OpenAPI</i>	3.0.X (3.1 non recommandée)

Les versions sont celles recommandées par les socles techniques à date de rédaction du document. Il conviendra d'utiliser les versions fournies par les socles (Fullstack) au moment du démarrage des développements.

5.2 Patterns généraux

5.2.1 Idempotence

Toutes les opérations effectuées par les apis, les batchs ou les traitements de messages sont implémentées de sorte qu'elles auront les mêmes effets qu'elles soient exécutées une ou plusieurs fois sur un même contexte.

5.2.2 Couplage lâche

Les principes de couplage lâche sont mis en place pour limiter les dépendances entre les composants. Le but est de faciliter :

- La réutilisabilité
- L'évolution indépendante des composants
- La testabilité
- Le déploiement et la supervision

D'un point de vue composants applicatifs ce pattern est mis en place via un découpage en micro-services, des contrats d'interfaces (api), l'utilisation de couche de médiation (ArchiMed) et d'un bus de messages (kafka). L'implémentation des services tient aussi compte de ce pattern via l'utilisation de l'injection de dépendance et un découpage en librairies réutilisables.

5.2.3 CQRS

Le CQRS (Command and Query Responsibility Segregation) est une architecture qui permet de séparer la lecture (Query) de l'écriture (Command). La partie écriture enregistre chaque modification, généralement sous forme d'évènement. Un traitement est ensuite opéré en asynchrone. La partie lecture requête simplement ces modèles afin de minimiser le requêtage de la base.

La mise en œuvre de l'architecture CQRS est recommandée mais ne fait pas encore l'objet de normes internes de mise en œuvre.

5.2.4 Coupe circuit

Afin d'assurer la résilience des services de l'application, entre eux et vis-à-vis des autres composants du SI, des mécanismes de coupe circuit sont implémentés via l'utilisation de librairie Resilience4j  #modif #3.2 et la mise en place de timeout. La mise en place de coupe circuits doit aussi être configurée au niveau de la gateway API ArchiMed.

5.2.5 Sûreté de fonctionnement

La conception doit permettre de garantir la fiabilité et la disponibilité du système dans les conditions de fonctionnement prévues (ressources disponible, volume de traitement).

La maintenabilité est prise en compte dans le DEX sous la forme de scénario de reprise adapté à chaque cas identifié afin de permettre une remise en fonctionnement fluide dans un état maîtrisé en cas de défaillance.

Les impacts et risques de sécurité ou de défaillance sur les autres composants du SI seront identifiés tout au long de la conception et consignés dans le DEX.

5.2.6 Impact

La solution ne doit pas dégrader le niveau de disponibilité des socles de développement, infrastructure, ni celui des autres services applicatifs avec lesquels elle interagit. Les éventuels impacts doivent être étudiés et toute dégradation fera l'objet d'un plan d'actions et d'actions correctives éventuelles associées.

5.2.7 Supervision et Observabilité des traitements

La solution doit fournir les données de télémétrie nécessaires à la mesure de ses performances et au contrôle de son bon fonctionnement. Ces informations devront être identifiables dans les logs de production. L'état courant du service doit être requêttable via des sondes.

5.2.8 Traçabilité des usages

La traçabilité fait référence à l'identification des actions métier effectuées sur le système. L'objectif est de permettre d'enregistrer les actions menées dans le SI pour disposer d'une preuve au sens juridique afin de couvrir des besoins législatifs (Obligation d'enregistrer une preuve pour la clôture d'un dossier incluant date, acteur responsable, ... par exemple) ou institutionnels.

Un service central de récupération des traces existe au sein du SI et doit être utilisé pour enregistrer les actions effectuées par les utilisateurs.

5.2.9 Contract first

La logique d'implémentation promue par fullstack est "contract first".

Tout développement d'un nouveau service doit commencer par l'écriture du contrat openapi suivi de la génération des classes DTO et interfaces de controllers.

Le fichier openapi sera déposé sur nexus afin d'être versionné et utilisé par les applications clientes.

Dans le cas d'une application clientes la librairie d'appel aux APIs devra être générée à partir du contrat d'interface récupéré depuis nexus.

5.2.10 Verrous optimistes

Plusieurs cas d'usage peuvent aboutir à des concurrences d'accès en écriture sur les données (Deux utilisateurs peuvent être amenés à effectuer une modification sur une même instance, un traitement automatique peut être amené à tenter d'effectuer une modification sur une instance en même temps qu'un utilisateur via l'IHM.)

Afin de garantir que le deuxième traitement ne vienne écraser le résultat du premier, et laisser le système dans un état potentiellement incohérent, toute modification effectuée par un traitement devra préalablement s'assurer que les données modifiées sont celles qui ont servi de référence au traitement (lock optimiste).

Les verrous optimistes lorsque nécessaires seront implémentés avec l'annotation @version JPA.

5.3 API

Les APIs fournies par les services utilisent le protocole REST. La norme adoptée pour les contrats REST est la spécification OpenApi 3.0 [Plus d'informations sur le site swagger.io](#). Elles doivent aussi suivre le standard OData ([Site OData](#)) en termes de syntaxe des filtres de recherche.

5.3.1 Généralités

5.3.1.1 Get simple

Chaque entité manipulée par APIs doit être récupérable de manière unitaire via un appel de type getByID.

5.3.1.2Création/Mise à jour

Dans le cas des APIs de création ou modification d'objets il est recommandé de privilégier une réponse comportant l'objet créé avec ses uuid afin de faciliter l'enchaînement des appels côté client.

5.3.1.3Listes

Les APIs retournant des listes doivent gérer une pagination avec un nombre d'enregistrement minimal par défaut pour empêcher le requêtage sans limite des tables. Les paramètres de pagination sont définis dans les normes.

5.3.1.3.1 Filtrage

L'utilisation du standard REST étant déjà encadrée par des normes Acoss, seule la syntaxe OData v4 des champs de type filter est retenue pour le projet.

Les API créées pour les besoins du projet devront être compatibles avec la syntaxe oData des champs filters en tenant compte des besoins projet pour ne pas implémenter l'ensemble des fonctions.

L'implémentation de ces opérateurs doit être suffisamment générique pour être mise à disposition des projets dans une librairie partagée.

OData (Open Data Protocol) définit un protocole d'interrogation et de mise à jour des données qui utilisent les protocoles Web existants. OData est un protocole basé sur REST qui repose sur les technologies standard comme HTTP, Atom/XML et JSON. Il est différent des autres services Web REST et constitue une solution uniforme pour la description des données et du modèle de données.

5.3.1.4Erreurs

Les erreurs remontées dans les réponses doivent comprendre les champs suivants :

- Exception ID – uuid de l'exception apparaissant aussi dans les logs
- Clé de message pour un traitement automatisé ou une traduction
- Texte du message

5.3.1.5Cache

5.3.1.5.1 Côté client

Les appels effectués sur d'autres services doivent prévoir un mécanisme de mise en cache adapté à la durée de validité des données afin de minimiser l'impact sur les services externe et améliorer les performances du client.

5.3.1.5.2 Côté serveur

Les services fournissant des apis doivent spécifier la durée de validité des données à l'aide des entêtes de cache. En particulier pour les apis destinées aux frontend.

5.3.2 Archimed

5.3.2.1Bundle

Un bundle Archimed est un ensemble de fichiers de configurations nécessaires au déploiement d'une api sur la Gateway. L'ensemble des fichiers nécessaires sont d'écrits dans un projet exemple (<http://gitlab.altair.recouv/java/gamme-sct/archimed/exemple-devops>)

Les principaux composant sont les suivants :

- Définition des apis
- Conf des règles de sécurité (policies)
- Définitions des clients

Tout composant mettant à disposition ou consommant des APIs doit comprendre un bundle Archimed et l'intégration continue associée.

5.3.2.2 Surcharge Swagger

Ces API sont mises à dispo via le système de médiation ArchiMed qui impose des contraintes supplémentaires à cette norme afin de garantir un traitement automatisé des contrats :

- Le contrat doit contenir une extension "x-backendArchimed" dans la section "info" qui a pour valeur le code de votre application, qui a été renseigné dans la grille d'autorisation Prisme
- Si le contrat contient un champ "schemes" (Swagger 2.0), le tableau doit contenir la valeur "http" ou bien le contrat ne doit pas contenir de champ "schemes"
- Toutes les opérations du contrat doivent avoir une valeur pour le champ "operationId", même si le service est non-sécurisé. Il est nécessaire d'avoir une valeur unique pour chaque opération, même pour des contrats différents
- Le chemin de base d'accès aux services, contenu dans le champ "basePath" doit avoir la forme suivante : `/<domaine>/<version>/<chemin>`
- Le `<domaine>` peut être simple (exemple : "archdemo") ou composé (exemple : "archimed/demo")
 - La `<version>` doit être de la forme vX.Y (exemples : v1, v1.2.3, v2.10)
 - Le `<chemin>` est facultatif, et peut être simple ou composé
 - Exemples : `/mon_appli/v1` ; `/mon/appli/v1/services/rest` ; `/app/archimed/v1.1/mes_proxy`

5.3.2.3 Gestion de la surcharge

5.3.2.3.1 Circuit breaker

Le bundle ArchiMed doit inclure une configuration de circuit breaker avec des seuils suffisamment élevés pour ne pas bloquer le trafic. Ces valeurs seront adaptées suite à une analyse du fonctionnement en prod.

5.3.2.3.2 Throttling

Le bundle ArchiMed doit inclure une configuration du trotting avec des seuils correspondant au fonctionnement attendu de l'application. Le throttling doit être configuré comme non bloquant.

5.3.3 Bouchons

Les apis fournies dans des bundles devront aussi être disponibles sous forme de bouchons.

La création de bouchons doit passer par un projet de type Fullstack dédié.

Le projet sera généré à partir de la définition d'APIs d'origine et des réponses statiques ou partiellement dynamiques construites dans les controllers.

Le projet devra être packagé sous forme d'image docker pour être intégré dans les environnements de test ou sur le poste développeur.

5.3.4 Consommation adaptative

La consommation d'API dans le cadre de traitements batch doit inclure un mécanisme de prise en compte du throttling et des quotas via les headers de réponse afin d'adapter la fréquence des appels à la capacité du service appelé et au quotas autorisés.

5.4 Batch

5.4.1 Framework

Le framework Spring Batch est utilisé pour implémenter les batch. Il fournit une implémentation mature, robuste et maintenable des couches, composants et services habituellement utilisés dans les systèmes mainframes pour créer des applications batch.

Le DSL Spring Batch permet de configurer à l'aide d'une API fluide le flux d'exécution de Jobsbatches composés d'étapes (Steps). Chaque Step est composé d'un ItemReader, d'un ItemProcessor et d'un ItemWriter. Un Job doit être lancé à l'aide d'un 'JobLauncher' et les métadonnées d'une instance de Job sont stockés dans un 'JobRepository' (schéma relationnel).

5.4.2 Planification et supervision de la planification

Il existe plusieurs solutions pour la planification des batch selon qu'ils s'agissent de traitement nationaux ou régionaux et selon les socles utilisés

5.4.2.1 Saturne

Saturne est l'application d'administration et d'ordonnancement nationale de la branche. Il doit être utilisé pour les batchs régionaux utilisant la base SNV2.

5.4.2.2 PFS

La planification de batchs dans le cadre des applications nationales conteneurisées déployées sur la PFS passe par une déclaration sous forme de configmap spécifique (contenant une déclaration de type job). La fréquence et les paramètres du job devront être fournis dans le DEX afin d'être déployé dans l'outil Automator par les équipes d'exploitation.

Afin de simplifier le travail en développement il est possible de construire un chart helm utilisant la valeur de l'environnement pour déployer en dev seulement un cronjob et utiliser une configmap sur les autres environnements.

5.4.2.3 Automator

La planification des batch doit passer par l'outil Automator qui est la solution retenue par la DSI pour l'ordonnancement et la planification des tâches informatiques. Il permet d'enchaîner des travaux selon un schéma de production à partir de critères variés : état d'une tâche précédente, événement particulier, action d'un utilisateur, calendrier et horaire. Il permet aussi d'enchaîner les travaux entre différentes applications.

5.5 Traitement des messages

Le bus de message utilisé par l'application est la solution Kafka. Le serveur Kafka est mutualisé et son utilisation est encadrée par une norme (§ Documents applicables).

Les développements Kafka doivent se faire en Contract First : La définition de la structure du message est donc à réaliser avant l'implémentation dans le code.

La configuration par défaut est

- Nombre de partitions : 1
- Facteur de répllication : 1
- Rétention : 7 jours

● #modif #3.2 #3.3

5.5.1 *Cas d'usage*

Les différents types décharges kafka sont en cours de standardisation selon 3 familles de publication : Ressource, Evènement fonctionnel, Request / Response.

Ce chapitre décrit les cas d'utilisation standard de kafka

5.5.1.1 *Evènement ressource*

Lorsque demandée par l'architecture, la publication d'évènement ressource doit suivre les principes suivants :

L'objectif est d'informer d'un changement de la Ressource (Create, Update, Delete).

Ce type de publication offre un spectre de couverture de besoins très large car il ne présume pas de l'usage fonctionnel qui sera fait par le consommateur. Toutefois, du point de vue des consommateurs, une action de filtrage sera probablement nécessaire.

Règles de déclenchement

La publication est pilotée à travers les opérations Create, Update, Delete

Publier tous les messages liés aux opérations Create, Update, Delete sur la ressource. Il ne faut pas présumer des consommateurs pour filtrer les messages envoyés.

S'assurer de la réalité d'une mise à jour (écarter le remplacement de la valeur d'une propriété par une valeur identique) pour minimiser la publication de message ressource.

Message de publication « Ressource »

En plus de l'identification de la ressource, pour faciliter le traitement du message consommé, on ajoutera :

Les informations sur la nature de l'évènement qui a mené au message : Opération réalisée (Create, Update, Delete)

Les informations de l'impact sur la ressource dans le cas où l'impact est une mise à jour : Liste des attributs de la ressource ayant été impactés sans leur valeur.

Dans le cas de la publication « Ressource » en tant que message « Response » de publication « Request / Response », l'identifiant du message « Request » qui a provoqué l'action.

5.5.2 *Kafka - Sécurisation et ACL*

5.5.2.1 *Sécurisation de la connexion à Kafka*

La connexion à Kafka doit se faire en Oauth2 et c'est recommandé d'utiliser la librairie Strimzi qui va gérer toute la mécanique Oauth2.

GitHub : <https://github.com/strimzi/strimzi-kafka-oauth>

Documentation : <https://strimzi.io/blog/2019/10/25/kafka-authentication-using-oauth-2.0/>

Voir les derniers exemples dans le projet Fullstack.

5.5.2.2 *Sécurisation de l'accès aux topics de l'application*

Il faut après sécuriser l'accès aux topics de l'application, donc il faut définir la liste des composants qui seront autorisés à y accéder ainsi que les opérations qui pourront être réalisées.

Impact bundle Archimed Exemple (se référer à la documentation [Sécuriser l'accès à vos topics](#) pour savoir comment configurer le bundle Archimed) :

```
{
  "id": "OAUTH-PRODUCER",
  "description": "Autorise l'application FULLSTACK-BE à produire des messages dans le topic monTopic",
  "match": {
    {
      "type": "ACL_OPERATION",
      "values": {
        "Production"
      }
    }
  },
  {
    "type": "ACL_TOPICS",
    "values": {
      "*.*.systemePorteur.monTopic1",
      "*.*.systemePorteur.monTopic2"
    }
  },
  {
    "type": "ACL_TRANSACTIONAL",
    "values": {
      "systemePorteur-tx-*"
    }
  },
  {
    "type": "ACL_USERS",
    "values": {
      "FULLSTACK-BE",
      "UN_AUTRE-BE"
    }
  }
},
{
  "id": "BT-OAUTH-CONSUMER",
  "description": "Autorise le consumer group monGroupeDeConsommateurs de l'application FULLSTACK-BE à lire les messages du topic monTopic",
  "match": {
    {
      "type": "ACL_OPERATION",
      "values": {
        "Consommation"
      }
    }
  },
  {
    "type": "ACL_TOPICS",
    "values": {
      "*.*.systemePorteur.monTopic",
      "*.*.systemePorteur.unAutreTopic"
    }
  },
  {
    "type": "ACL_CONSUMERS_GROUPS",
    "values": {
      "*.*.systemePorteur.monTopicListener",
      "*.*.systemePorteur.unAutreTopicListener"
    }
  },
  {
    "type": "ACL_USERS",
    "values": {
      "FULLSTACK-BE",
      "UN_AUTRE-BE"
    }
  }
}
}
```

5.5.2.3 Sécurisation de l'accès aux topics transverses

● #modif #3.3

Les topics transverses comme debezium SNV2 ou topics techniques sont gérés par la MOE Archimed.
Les demande d'accès doivent leur être adressées via redmine avec le détail concernant le consumer et les topics utilisés.

5.5.3 Nomenclature des topics

● #modif

La nomenclature des topics est encadrée par la norme Kafka (NME_TOPIC_3 : Nommage des topics) voir exigence "Bus de messages - Recommandations techniques/SDED - Normes Kafka".

● #modif #3.2

Les noms de TOPIC doivent être variabilisés au niveau des applications pour être injectés via des variables d'environnement (Castor, token, values).

5.5.4 Partitionnement

Un partitionnement adapté à chaque topic est à définir pour répondre aux problématiques de performance et de concurrence de consommation.

5.5.5 Structure des messages

5.5.5.1 Format

Les messages sont au format binaire Avro.

L'application doit utiliser une sous-classe de `KafkaAvroDeserializer` (`KafkaAvroDeserializer Custom`) pour transformer ce format binaire en text/JSON.

5.5.5.2 Taille

Les messages transmis via kafka à destination de l'ensemble du SI doivent rester légers (de type DTO). La logique étant de transmettre l'évènement en laissant à la charge du consommateur de récupérer des données à jour au moment du traitement. Dans les cas où la donnée est très variable il faut privilégier le fait de transmettre seulement l'id de l'objet et les valeurs modifiées.

Il est possible de faire exception à cette règle pour un message interne à l'application transitant sur un topic intra. Ce type de message pourra reprendre l'ensemble des champs de l'objet métier s'il est consommé rapidement par un autre service interne et que la durée de rétention nécessaire est courte.

5.5.5.3 Gestion des erreurs de désérialisation

La méthode `deserialize` est surchargée pour gérer les erreurs de la manière suivante :

- Appeler la méthode `deserialize` de la classe mère `KafkaAvroDeserializer`.
- En cas d'erreur (exception), générer les logs suivants :
 - ERROR : Type et message de l'exception
 - INFO : Détails : topic, contenu en mode String et en mode binaire (tableau d'octets)
- Retourner un enregistrement (`GenericData.Record`) contenant un seul champ ERROR avec le type et message de l'exception.

Le traitement ignorera ensuite ces messages, et attend le message suivant.

5.5.6 Consommation

La stratégie de consommation doit être décrite dans les STD et inclure les choix en termes de démarrage, mécanisme de requêtage, buffers, commit, lag

5.5.6.1 Démarrage et reprise

Les agents consommateurs de messages Kafka doivent disposer d'un mécanisme pour spécifier l'offset du premier message à traiter afin de supporter les cas particuliers de remise en service suite à une défaillance.

5.5.6.2 Logiques de streaming

Des mécanismes de requêtage des stream kafka adaptés à chaque traitement devront être mis en place selon les besoins pour grouper les évènements.

- Reduction
- Agrégation
- Join
- Windows

5.5.7 Mise à l'échelle

Les composants de type consommateur kafka doivent pouvoir être mis à l'échelle si le topic est partitionné.

La conception doit permettre le fonctionnement en mode parallélisé. ● #modif #3.3

5.5.8 Commit

● #modif

La stratégie de commit doit permettre un bon ratio entre performance et sécurisation de l'état. Les cas d'échec de consommation doivent être identifiés. Le comportement de l'applicatif suite à un arrêt normal ou forcé ayant causé une double consommation doit être décrit.

5.5.9 Evolutions en cours

● #modif

Des évolutions sont en cours de développement pour améliorer la qualité des dates dans certains messages :

- Debezium – les champs correspondant à des colonnes de table contenant un timestamp vont être modifiés pour tenir compte de la timezone de la base de donnée concerné (cas des dom tom etc..)
- Cycle de vie SNV2 – Les dates des messages vont être modifiées pour inclure la timezone de la base de données

Ces travaux en cours devront être pris en compte lors de leur mise à disposition.

5.6 Stockage S3

Le stockage de fichier sur S3 doit répondre à des conventions de nommage décrites ici : <http://exploit-wikijs.cnp.recouv.fr/documentations/transfert-de-fichiers/Bucket-s3.md> et ici <https://readthedocs.cnp.recouv/docs/norme-gelar/fr/latest/search.html?q=S3>

On considère 3 types de buckets à créer selon les usages.

partage

Nom : prod-appli-partage

Propriétaire : pfs-prod-transfert

Utilisation : déposer les fichiers produits par mes différents batchs (ou jobs pour k8s)

Qui peut lire : uniquement les applications qu'on autorise à la création du bucket.

Qui peut écrire : uniquement mon application.

reception

Nom : prod-appli-reception

Propriétaire : pfs-prod-transfert

Utilisation : récupérer les fichiers produits par le legacy (comprendre : par une application non compatible S3 aujourd'hui ; transmis par le proxy de transfert) pour être traités par mes différents batchs (ou jobs pour k8s)

Qui peut lire : uniquement mon application.

Qui peut écrire : uniquement le proxy de transfert.

travail

Nom : prod-appli-travail

Propriétaire : l'utilisateur s3 de mon application (nommé généralement prod-appli).

Utilisation : stocker les fichiers de travail de mon application qui ne sont pas à partager

Qui peut lire : uniquement mon application.

Qui peut écrire : uniquement mon application.

La création du bucket doit passer par le subchart s3 (<https://gitlab.cnp.recouv/k8s/chart/create-bucket>)

5.7 Logs

● #modif

Les logs de la solution doivent respecter la Norme QELAR et celles de Fullstack Back et front-end

5.7.1 Contenu

Les logs techniques de la solution ne doivent pas contenir d'informations personnelles tel que nom, prénom, nir, etc ...Ces informations peuvent être enregistrées sur les canaux de log de traçabilité : empreinte.

Les logs doivent contenir un maximum d'informations permettant de reconstituer le contexte. Lorsqu'ils sont disponibles les champs suivants sont ajouté dans le log :

- Identifiant de corrélation de la transaction
- Code UR
- Type et UUID de l'entité manipulée
- UUID de l'exception
- Stack trace
- Topic
- Client_id
- Checksum du token

Lorsque c'est applicable, ces informations doivent être séparés du message et stockées dans des champs dédiés.

Le message de log doit être court et précis afin de faciliter les recherches dans des outils de type kibana. Dans une même fonction on ne doit pas retrouver 2 fois le même message de log sans moyen de les distinguer.

5.7.2 Emplacement

L'emplacement des logs est défini par la variable LOG_PATH.
La valeur standard est /log (répertoire de log à l'intérieur du conteneur).

5.7.3 Niveaux de logs

Le niveau de log est paramétrable de la façon suivante :

- **FATAL** : erreurs techniques empêchant le lancement de la brique
- **CRITICAL** : Service dégradé
- **ERROR** : erreurs fonctionnelle sérieuses empêchant le traitement d'un message (abandon du traitement)
- **WARN** : erreur fonctionnelle permettant la poursuite du traitement du message
- **INFO** : niveau de log nominal (quelques lignes par événement). Ce niveau est activé par défaut.
- **DEBUG** : niveau de log détaillé, contenant notamment le contenu de chaque message et des informations précises sur le traitement (volumineux)
- **TRACE** : niveau de log le plus détaillé, avec l'intégralité des échanges, les headers des messages et toutes les informations techniques (très volumineux)

5.7.4 Configuration

Le framework de log est **logback**.

Ce dernier est configuré pour générer deux fichiers :

- Le fichier « raf_<composant>.log » contenant les traces de tous les niveaux. Son niveau par défaut est INFO.
- Le fichier « raf_<composant>_error.log » contenant les traces de niveau ERROR, CRITICAL et FATAL.

La rotation des logs est gérée ainsi :

- 31 jours de rétention, avec un maximum de 20 Go. Au-delà, les anciens fichiers sont supprimés.
- Les fichiers sont archivés (compressés) chaque jour.

5.8 Base de données

5.8.1 Généralités

5.8.1.1 Droits d'accès

On identifie 3 rôles pour l'accès aux bases de données :

- RAF_DDL_<nomde la base> - Pour les modifications de schéma
- RAF_RW_<nomde la base> - Pour le fonctionnement normal de l'application
- RAF_RO_<nomde la base> - Pour les accès statistiques ou de diagnostique

Dans le cas des plateformes SX la gestion des tables est gérée par le socle. Un utilisateur est déjà en place pour la création des schémas : sysdba.

5.8.1.2 Convention de nommage

On distinguera les préfixes suivants pour les colonnes :

- "ID" pour les colonnes de primary key
- "FK" pour les clés étrangères pointants vers une autre table
- "REF" pour les références à des id provenant de l'extérieur de la solution
- "CODE" pour les id fonctionnels interprétables humainement. Ces derniers devront contenir du texte en spinal case

5.8.1.3 Meta données

Les tables doivent comporter les métas donnés suivantes :

- Date de création
- Id de corrélation de la transaction de création
- Date de mise à jour
- ID de corrélation de la transaction de mise à jour

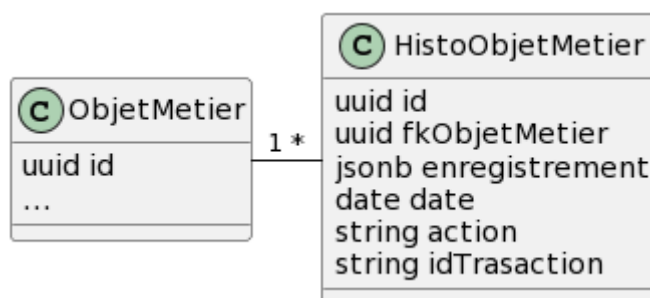
En cas d'application d'une stratégie de soft delete

- Date de suppression
- ID de corrélation de la transaction de suppression

5.8.1.4 Historisation

Dans certaines situation une historisation technique des changements effectués sur les données est requise.

Le modèle suivant est retenu pour conserver les valeurs successives des enregistrements à des fin de restauration ou d'audit :



Selon les cas l'historisation se fera via la librairie Envers.

La solution a utiliser devra être confirmée avec la MOE.

● #modif #3.3

5.8.1.5 Stratégie de suppression

La stratégie de suppression par défaut est le soft delete. Elle doit être remise en cause si les volumétries ne le permettent pas. Afin de pouvoir nettoyer la base de données des enregistrements supprimés une requête SQL sera fournie dans les sources et référencée dans les documents d'exploitation.

5.8.2 BDD SNV2 SX

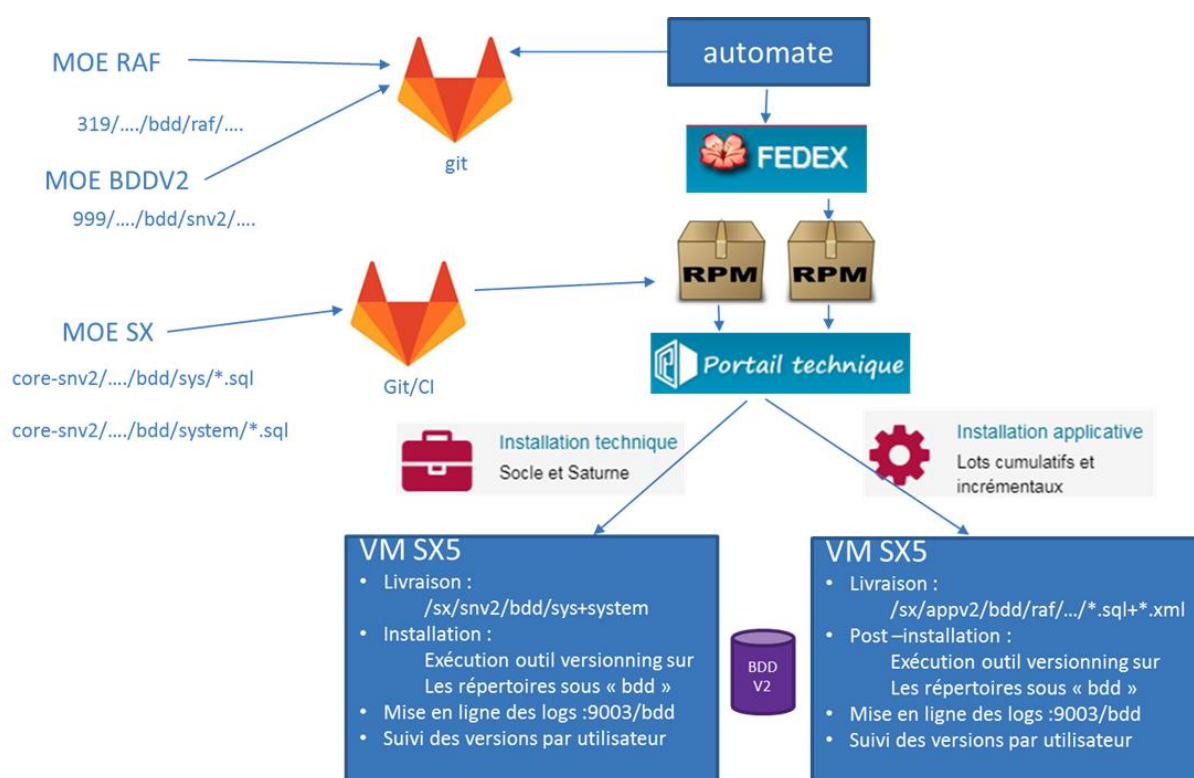
5.8.2.1 SGBD

La solution doit être compatible avec le SGBD : **PostgreSQL**

Les développements ne devront pas utiliser de fonctions spécifiques aux implémentations des éditeurs afin de pouvoir être réutilisés sans contraintes dans d'autres implémentations.

5.8.2.2 Evolution du schéma

Toute création, modification ou suppression touchant au schéma rénové de la base de données SNV2 devra se faire à travers l'outil **Liquibase** selon les préconisations de la documentation sx5 fournie en référence.



1 - Principe Liquibase SNV2

Les fichiers liquibase seront au format UTF8 seront stockés dans le dossier `/sx/appv2/bdd/raf-parcours` du dépôt cntx.

5.8.2.3 Cas des restaurations

La base de données peut être restaurée en cas d'erreur lors d'un processus batch existant ou lors d'une demande de l'Agent Comptable après le redémarrage du transactionnel.

La solution doit considérer ces possibles restaurations et le cas échéant retrouver un état fonctionnel cohérent.

5.8.3 BDD Nationales

Les composants nationaux de la solution sont basés sur une base de données unique PostgreSQL.

La base de données nationale doit être déployée sur des serveurs postgres mutualisés pour la MOE et hébergés sur Hawaï.

Le cluster PostgreSQL doit être configuré avec les options de haute disponibilité.

Une procédure et un script Shell pour permettre de récupérer des backups devront être fournis à la production

5.9 IHM

● #modif

Les IHM de la solution devront se baser sur le projet « Fullstack SPA » qui fournit le modèle d'une application front.

Les technologies à utiliser sont :

- Typescript
- Angular
- Bootstrap
- AgGrid
- Lib de communication UCN
- Lib authentification

La charte graphique à respecter est décrite à travers des exemple de composants fournis dans l'application fullstack-spa ainsi que les documents d'ergonomie (<https://kxzkx5.axshare.com/>).

5.9.1 Intégration PORA

● #modif

Si l'application est destinée à être intégrée dans PORA elle doit uniquement produire l'ihm correspondant à son périmètre. La navigation en haut, le menu de gauche et les fenêtres de connexion sont gérées par le portail. L'IHM de l'application doit être capable de s'afficher dans un iframe correspondant à la zone de l'application.

L'ouverture de l'application par le portail est immédiatement suivie par l'envoi d'un message par le portail via la lib de communication. Une réponse à ce message doit être retournée par l'application pour confirmer sa bonne ouverture. D'autre messages peuvent ensuite être transmis par le portail afin de transférer des informations partagées ou des actions de l'utilisateur (clic dans les menus).

L'ensemble des messages et leurs contenus devront suivre les spécifications décrites dans le dossier d'intégration PORA.

L'IHM doit être accessible en HTTPS

5.9.2 Intégration BIPE

● #modif

Lorsque applicable les IHM doivent implémenter un descripteur d'écran format BIPE et la gestion des évènement associés afin de permettre l'interfaçage avec les autres applications du SI.

Voir la documentation dans l'exigence : "Frontend - Utilisation de BIPE".

5.9.3 Consommation d'APIs

Le code client d'API (DTO, services...) doit être généré à l'aide de la librairie swagger-gen incluse dans le projet fullstack afin de pouvoir être facilement mis à jour lors des évolutions du contrat d'interface.

Les appels d'APIs relatif au chargement des données pour l'affichage doivent être organisés de sorte à optimiser la fluidité du rendu et de la navigation.

L'application doit respecter les règles de cache fournies par les headers HTTP des réponses d'API.

La manipulation (agrégation, consolidation ...) de données issues des API soit être effectuée dans un service dédié permettant de partager cette logique entre plusieurs écrans ou composants utilisant ces informations.

5.9.4 Stockage

Le stockage d'informations dans le navigateur doit être limité et les cas d'utilisation clairement documentés dans les STD. Il ne doit pas contenir de données sensibles ou à caractère personnel.

Le LocalStorage doit être limité à des préférences utilisateur non persistées coté serveur.

5.9.5 Composants réutilisables

Les éléments d'interface spécifiques à l'application mais utilisés sur plusieurs écrans doivent être développés sous forme de composants.

5.9.6 Format des images

Le format recommandé des images est SVG.

5.9.7 Nettoyage des exemples fullstack

L'application fullstack-spa contient de nombreux exemples ainsi que du code permettant de simuler un contexte de portail. Ces derniers doivent être retirés de l'application avant toute livraison et le code restant doit concerner le périmètre de l'application.

5.9.8 Bonnes pratiques

Le code doit respecter les bonnes pratiques relatives à un projet angular.

- Typage
- Utilisation d'observables avec les bons opérateurs et souscriptions
- Services et opérateurs Tree-Shakable
- LazyLoading
- TrackBy
- ...

5.10 Conteneurisation

La solution utilise des conteneurs pour le déploiement.

5.10.1 Construction des images

Les images docker doivent partir d'une image disponible sur le dépôt harbor privé de l'UCN.

L'utilisation d'une image non disponible devra faire l'objet d'une demande d'ajout.

La construction des images docker des projets java doit se baser sur les préconisations de Fullstack mais peut être adaptée ou améliorée si le projet le nécessite.

5.10.2 Plateformes cibles

Les modalités de mise en œuvre dépendent des plateformes cible elles-mêmes déterminées par le type de composant.

Les conteneurs régionaux sont déployés sur les plateformes SX, les composants nationaux sur la PFS.

5.10.2.1 SX

*La mise en œuvre de conteneurs sur les plateformes SX doit suivre le document « **SNV2 – Container Webapp Régional** » qui encadre la construction des images et leur paramétrage dans les lots V2 et les ressources disponibles au niveau du conteneur.*

5.10.2.1.1 Exécution

Le déploiement des conteneurs est géré par le socle SX de manière automatique.

Les conteneurs sont administrables au niveau exploitation via des variables d'environnements et des fichiers de configurations. En particulier la variable POD_LOCATION permet de choisir d'exécuter les conteneurs sur la machine SX ou sur le cluster K8S associé.

5.10.2.1.1.1 K8S

Par défaut les conteneurs SX sont déployés sur un cluster k8s dédié.

5.10.2.1.1.2 Local

Le déploiement local de containers n'est pas la cible en production. Il peut être utilisé sur les environnements de dev afin de faciliter certaines analyses.

L'exécution forcée sur la machine SX est possible en cas de besoin très spécifiques irrésolvables sur k8s et devra faire l'objet d'une étude par l'équipe socle SX.

5.10.2.2 PFS

5.10.2.2.1 Helm

Le déploiement de charges de travail sur les clusters de la PFS doit passer par des chart HELM.

*Leur définition est stockée dans un projet git sur la base du **dépôt de référence** : [sddos / Innovation et transformation digitale / procedure_deploiement_pfs / depotReference · GitLab \(altair.recouv\)](#)*

Les charts devront variabiliser toutes les paramètres de l'application pouvant dépendre de l'environnement.

Il est nécessaire de séparer la version de l'application de la version du chart.

En effet la version applicative est portée par le tag de l'image docker par exemple alors que la version du chart ne doit dépendre que de la version de ce chart. Une modification du chart n'entraîne pas forcément une modification de la version applicative.

La préconisation est de fixer la version applicative dans le chart (surtout pour les versions qui sortent du cycle de développement) donc un changement de version applicative entraîne forcément une modification de la version du chart. Cela permet d'assurer une traçabilité et un déploiement cohérent notamment dans le cas de la livraison de plusieurs services qui ne possèdent pas un couplage lâche entre eux. C'est alors le chart qui définira les dépendances.

Le chart doit utiliser au maximum les opérateurs disponibles sur la PFS pour répondre aux besoins techniques de la solution (ex : base de données, composant réseau, elk).

Le chart doit aussi clairement définir les ressources nécessaires et maximale pour chaque pod.

5.10.2.2.2 Operateurs

Lorsqu'un opérateur est disponible sur la PFS il doit être privilégié pour le déploiement des services qu'il gère.

Les opérateurs actuellement disponibles sont :

- Kasten (backup)
- CrunchyData (cluster PostgreSQL)

5.10.2.2.3 Registry

Le registre Harbor est utilisé pour le stockage des images et des charts HELM. Il existe un registry de dev et un de prod. L'utilisation du Jenkins est nécessaire pour builder et déposer les images sur Harbor.

Nexus est aussi utilisé pour le stockage des images de builds.

5.10.2.2.4 Archimed, keystore

Les applications clientes d'un service ou exposant un service sur Archimed doivent comporter des éléments tels que :

- Les magasins de certificats (Keystore/Truststore) qui devront contenir les certificats des autorités racines (AC) de l'acoss et qui sont protégés par un mot de passe correspondant au clientSecret de l'application
- La/Les paire(s) de clés publiques/privées nécessaires pour s'authentifier lors des appels à la gateway

Pour ce faire le dépôt de référence contient 2 scripts appelés depuis le script deploy.sh et qui sont chargés de créer le magasin de clés et de récupérer la/les paires de clés.

5.10.2.2.5 Communication

5.10.2.2.5.1 API

L'accès aux API d'un pod du même namespace peut être en direct sur le service. Tout appel à une API hébergée sur un autre namespace devra être faite via Archimed.

5.10.2.2.5.2 NetworkPolicies

Des NetworkPolicies restreignant l'accès au niveau du namespace doivent être mise en place. Celles-ci sont basées sur le composant Cilium et devront donc être définies via la CRD CiliumNetworkPolicies.

Elles doivent clairement identifier les différents flux nécessaires à l'application.

5.10.2.2.5.3 Istio

Un sidecar Istio doit être configuré pour gérer le routage du trafic entre les services.

5.10.2.2.5.4 Accès externe

Les flux entre les clusters PFS et le legacy sont coupés par défaut. En sortie de PFS l'application doit donc utiliser la gateway Archimed ou le Trafic manager (BigIP / F5).

Les flux HTTP (Web services...) passent par Archimed, les autres (jdbc, kafka,...) doivent passer par le Trafic manager (BigIP / F5).

Une demande de configuration doit être faite pour la partie BigIP.

Au sein de la PFS le routage est à configurer via Istio.

5.10.2.2.6 Vault

L'outil Vault doit être utilisé pour stocker toutes les variables nécessitant d'être chiffrées. Des exemples d'initialisation de services PFS utilisant Vault peuvent être fournis par l'équipe responsable du socle.

5.10.2.2.7 Exposition des métriques

La PFS propose en standard 2 outils :

- prometheus qui va venir récupérer ("scrapper") les métriques exposées à intervalle régulier
- grafana qui va permettre d'exposer un tableau de bord constitué de graphique réalisés à l'aide de ces métriques

*Afin que prometheus vienne récupérer les métriques à intervalle régulier la solution doit mettre en place un objet **ServiceMonitor** pour fournir à minima :*

- le service
- Le chemin (url) sur lequel sont exposées les métriques
- le port

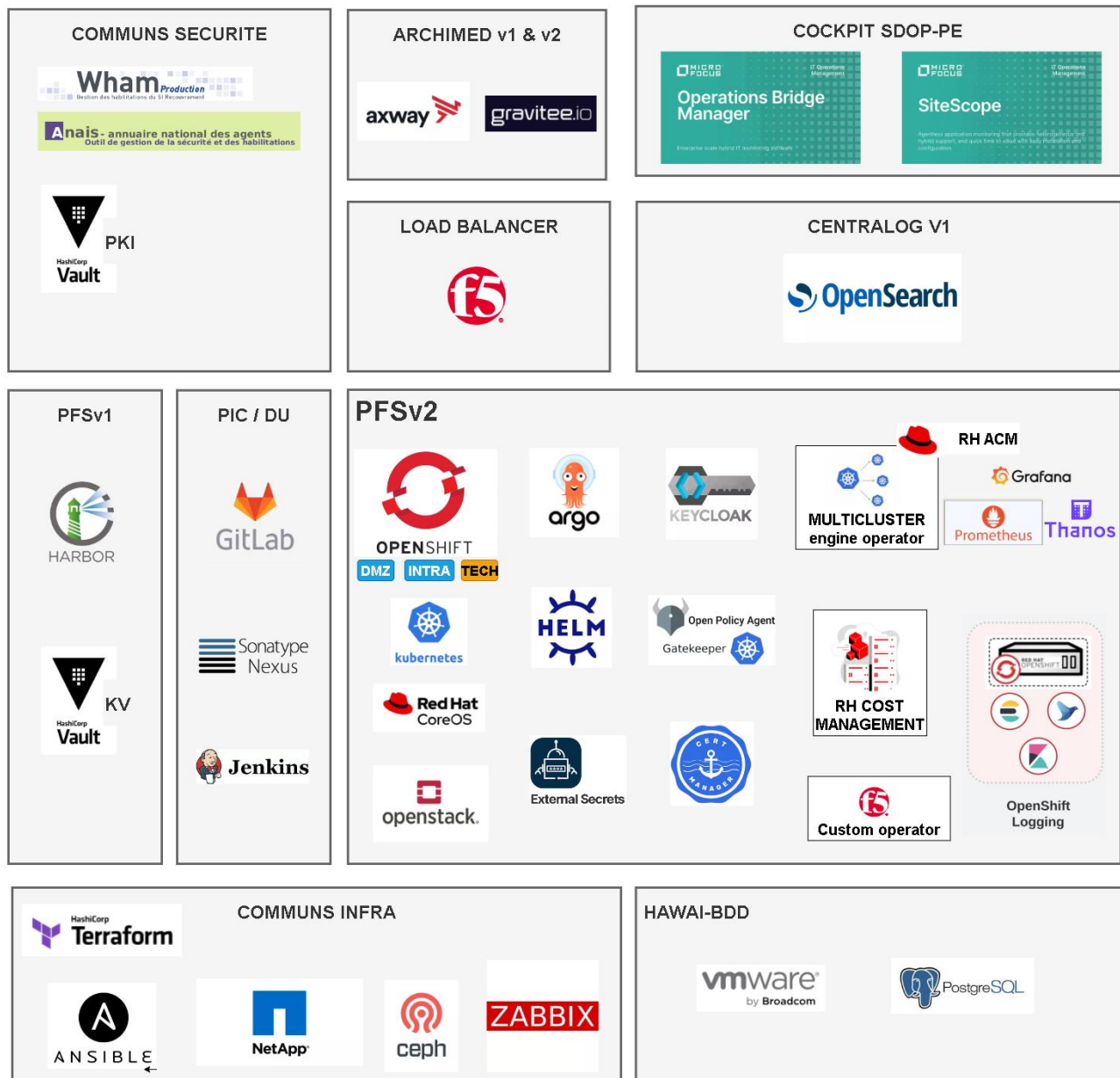
5.10.2.3 PFS V2

● #modif #3.3

La plateforme PFS v2 est décrite dans la documentation confluence suivante : [PFSv2 - Documentation utilisateurs - PFSv2 - Documentation utilisateurs - Confluence DSI Acoess](#)

Elle est composée des briques et technologies suivantes :

Technologies constituant le PaaS incluant la PFSv2
Vue en Janvier 2025



PFSv2

Technologies assurant le fonctionnement des applications métiers - Dec. 2024

RESEAU & MEDIATION



gravitee.io

axway
API Manager

PKI

axway
AUTOMATOR

SECURITE & ORHCESTR.

SERVERS & RUNTIME



spring boot

NGINX

openstack.



OPENSHIFT



kubernetes

COMPUTE

SGBD



PostgreSQL



NetApp



ceph

STOCKAGE

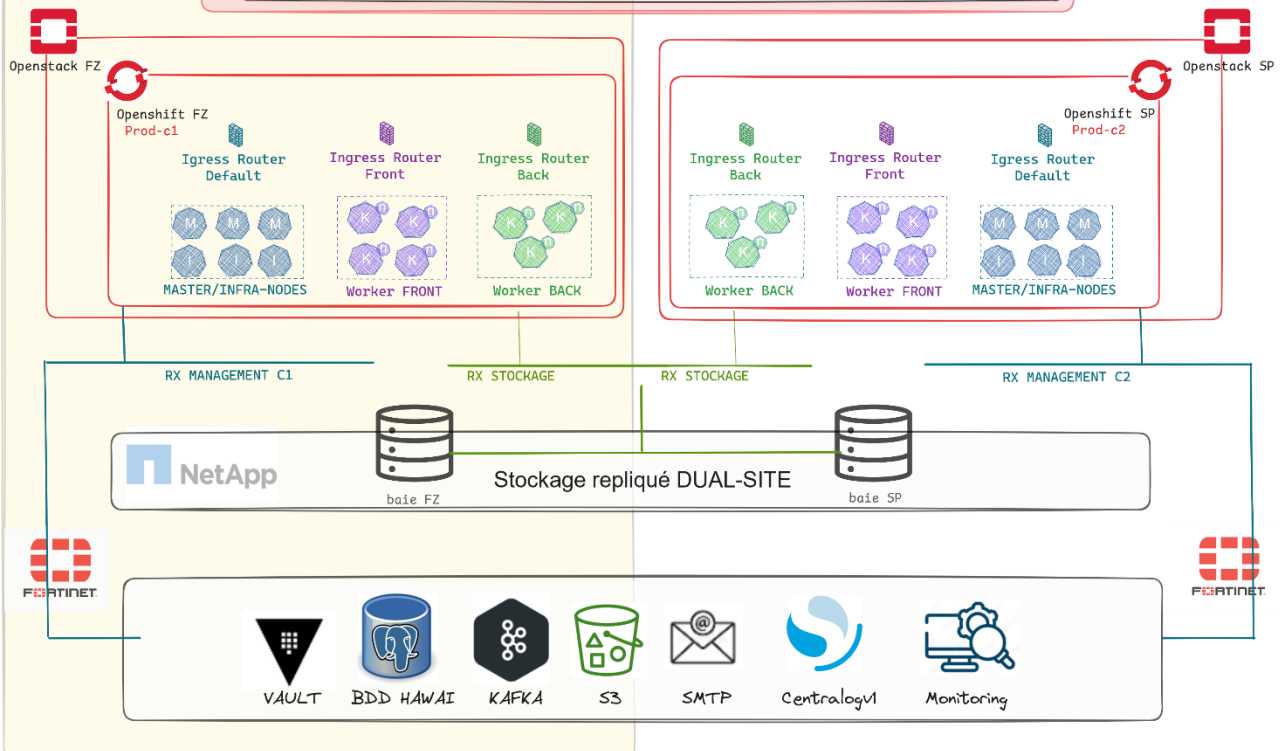
Datacenter - ST PRIEST

Datacenter - FEYZIN

ARCHIMED v2: Gravitee

ARCHIMED v1: Axway

LoadBalancer : F5 BACK VIP ADMIN C1 VIP APPS BACK VIP ADMIN C2
LoadBalancer : F5 FRONT VIP APPS FRONT



5.10.3 Particularités sur PFS de dev et staging

● #modif Les charts Helm de déploiement en PFS peuvent contenir des spécificités pour les environnements de dev et staging uniquement via un chart séparé.

Les bases de données doivent être exposée via un service afin de permettre l'accès via des clients lourds (dbeaver, pgadmin ...) lors des phases de validation.

Les jobs récurrents déployés en prod via des config map utilisées par Automator peuvent être configurés en dev et staging sous forme de CronJob classiques afin de simuler le comportement de prod sans Automator.

5.11 Tests

Les tests backend ou frontend doivent suivre les recommandations des documents :

- Norme automatisation des tests (<https://readthedocs.cnp.recouv/docs/fullstack/fr/latest/normes/testsauto/index.html>)
- Test E2E ([Guide Tests de bout en bout \(E2E\) — Documentation Fullstack 1.0.0 \(cnp.recouv\)](#))

Aucune méthode de test (ou une partie) ne doit être désactivée.

5.12 Divers

5.12.1 Conventions de nommage

5.12.1.1 Langue utilisée

Le nommage des variables, classes et répertoires est généralement en français mais afin de faciliter la compréhension des concepts de programmation et l'association avec les librairies ou frameworks certains éléments doivent rester en anglais.

Français :

- Variables métier
- Répertoires et fichiers métier

Anglais :

- Concepts et patterns de programmation (mapper, validator, ...)
- Éléments hérités des frameworks et librairies

5.12.1.2 Méthodes de test

Pour une meilleure lisibilité, les noms des méthodes de test doivent suivre le template suivant : testMethodeATesterEtat, où MethodeATester est le nom de la méthode à tester. Etat = KO, OK ou autre état spécifique.

Exemple : pour tester executerEnchainement(), deux méthodes de test sont à créer : testExecuterEnchainementKO() et testExecuterEnchainementOK()

5.12.2 Particularités sur environnement de dev et incubateur

● #modif

- L'accès aux bases de données postgresql de PFS peut être facilité par l'ajout d'un loadbalancer utilisant un port 6xxxx. Ce loadbalancer doit être créé manuellement ou conditionné dans le chart helm par une variable "environnement" afin de garantir qu'il ne sera pas déployé en prod.

- Les CronJob K8s peuvent être configurés en dev mais ils doivent être conditionnés pour utiliser une configuration spécifique automator sur les autres environnements

5.12.3 *Commentaires et lisibilité du code*

● #modif

Lorsque le nom de la méthode ou la variable est explicite, il est inutile de rajouter un commentaire par-dessus. Cela nuirait à la lisibilité du code.

Exemple : au lieu de ce code :

```
/** service. */  
@Autowired  
private PersonneService service;  
/** mapper. */  
@Autowired  
private PersonneMapper mapper;
```

Nous devrions avoir ceci :

```
@Autowired  
private PersonneService personneService;  
  
@Autowired  
private PersonneMapper personneMapper;
```

6 Supervision

6.1 Sondes

6.1.1 Principe général

Le test de vie est implémenté en utilisant la couche de supervision intégrée à Spring : Spring Boot Actuator.

Le mécanisme d'auto-configuration de Spring Boot activera automatiquement la supervision, incluant un test de vie. Ce service se présente sous la forme d'un endpoint accessible en http incluant notamment une méthode « /health », permettant de vérifier la bonne santé de l'application.

Le service est embarqué dans le même process Java que les services Spring Integration, dont il partage les ressources et la configuration.

6.1.1.1 Configuration

Pour respecter la norme en vigueur à l'ACOSS, le endpoint du service actuator est reconfiguré pour pointer sur le chemin « **/management** ».

Les réponses du HealthCheck sont normalisées :

- UP = service démarré et prêt à rendre les services applicatifs
- DOWN = service indisponible

La surveillance des data sources de l'application est activée.

Enfin, le timeout pour le polling des messages Kafka est également configuré.

```
management.endpoints.web.base-path=/management
management.endpoint.health.kafka.timeout=5000
management.health.db.enabled=true
management.health.defaults.enabled=false
```

6.1.2 Tests de vie personnalisés

Par défaut, le service Actuator se limite à un test de connectivité réseau (port ouvert). Cependant, pour s'assurer que l'application est pleinement fonctionnelle, des tests supplémentaires sont nécessaires. Ceci est réalisé par le biais de tests de vie personnalisés.

6.1.2.1.1 Vérification de la connectivité avec la base

Actuator fournit de base un test pour les data sources. Il suffit d'activer cette fonctionnalité dans le fichier de configuration de l'application.

La connectivité avec la base est ainsi contrôlée puisque le service Spring Integration et Actuator partagent la même configuration (et les mêmes data sources).

6.1.2.1.2 Vérification de la connectivité Kafka

Pour la vérification de la connectivité Kafka, il n'existe pas de classe « HealthIndicator » native. L'interface sera donc implémentée et sa méthode « health » sera chargée de se connecter aux topics utilisés par le module.

6.1.2.1.3 Resultat JSON

Lorsque le résultat est UP, le JSON retourné est de la forme suivante :

```
{
  "status": "UP",
  "components": {
    "db": {
      "status": "UP",
      "details": {
        "database": "Postgre",
        "result": "Hello",
        "validationQuery": "SELECT 'Hello' from DUAL"
      }
    },
    "kafka": {
      "status": "UP",
      "details": {
        "functional": {
          "topic": "GDC.SITUATION_COTISANT.APRES.TERME",
          "status": "UP",
          "info": "0 messages available"
        },
        "technical": {
          "topic": "UR837M",
          "status": "UP",
          "info": "0 messages available"
        }
      }
    }
  }
}
```

Le statut global est retourné à la racine du JSON (champ status).

L'état de la base de données est dans components.db.

L'état de Kafka est dans components.kafka.status. Les détails pour les topics fonctionnels et techniques sont dans components.kafka.details.fonctionnal et components.kafka.details.technical respectivement.

Le champ info contient soit le nombre de messages disponibles dans le topic quand le topic est UP ou sinon, la cause de l'erreur.

En cas d'erreur, le retour est de la forme suivante :

```
{
  "status": "DOWN",
  "components": {
    "db": {
      "status": "DOWN",
      "details": {
        "error": "org.springframework.jdbc.CannotGetJdbcConnectionException: Failed to obtain JDBC Connection; nested exception is java.sql.SQLTransientConnectionException: HikariPool-1 - Connection is not available, request timed out after 30010ms."
      }
    },
    "kafka": {
      "status": "DOWN",
      "details": {
        "functional": {
          "topic": "GDC.SITUATION_COTISANT.APRES.TERME",
          "status": "DOWN",
          "info": "Topic GDC.SITUATION_COTISANT.APRES.TERME unreachable"
        },
        "technical": {
          "topic": "cdctest",
          "status": "DOWN",
          "info": "Topic cdctest unreachable"
        }
      }
    }
  }
}
```

6.2 Métriques

Un certain nombre de métrique de base sont interprétables à partir des logs INFO de l'application. Les healthcheck actuator fournissent aussi des informations en standard.

Les métriques spécifiques de l'application devront être mise à disposition via micrometer.

Elles devront ensuite être fournies dans le DEX pour décrire le monitoring associé.

Les indicateurs suivants ont déjà été identifiés :

- Volume de messages
- Temps de traitement message
- Volume de données du batch
- Durée du batch

7 Sécurité

7.1.1 Sécurité des développements bonnes pratiques

La solution doit respecter les bonnes pratiques suivantes pour réaliser les développements demandés :

- Mettre en place des mesures contre le top10 de l'OWASP version 2021 ● #modif
- Les bonnes pratiques de codage issues de l'Owasp (document Acooss en annexe)
- Le kit des bonnes pratiques dispensé par la CNIL : <https://www.cnil.fr/fr/developpeurs-la-cnil-met-en-ligne-un-kit-de-bonnes-pratiques>
- Le guide d'hygiène de l'Anssi : <https://www.ssi.gouv.fr/guide/guide-dhygiene-informatique/>

7.1.2 Composants utilisés et vulnérabilités

Les composants et librairies utilisés sont éprouvés et garantis sans vulnérabilités à la livraison.

7.1.3 Charte d'application

La sécurité des développements doit être assurée conformément à l'état de l'art et aux règles des normes de sécurité fournies.

Ci-dessous une liste (non exhaustive) de règles applicables :

- Utiliser obligatoirement des outils permettant de minimiser les erreurs introduites durant le développement (outils d'analyse statique de code, utilisation de bibliothèques réputées pour leur sécurité, etc.).
- Produire une documentation technique et dans le code décrivant l'implantation des protections développées (gestion de l'authentification, stockage des mots de passe, gestion des droits, chiffrement, etc.) ;
- Respecter les normes de développement sécurisé, qu'elles soient propres au développeur, publiques ou propres à l'ACOSS ;
- S'obliger à corriger, au titre de la garantie et dans un temps raisonnable, les vulnérabilités introduites durant le développement qui lui sont remontées, en incluant automatiquement les corrections des autres occurrences des mêmes erreurs de programmation.
- Assurer un contrôle rigoureux des données d'entrées utilisateurs ;
- Sécuriser les accès aux fonctions d'administration ;
- Respecter le principe du moindre privilège ;
- Proscrire les mots de passe en dur dans le code ;
- Mettre en œuvre d'une gestion efficace des erreurs.

Pour la mise en œuvre de technologies web, les développements s'appuieront sur les recommandations de l'OWASP (Open Web Application Security Project) et notamment l'ASVS.

7.1.4 Contrôle des dépendances

Il est nécessaire d'intégrer aux pipelines de build une étape de contrôle des dépendances à l'aide de « Dependency Check ». Les niveaux attendus sont décrits dans la partie livraison.

8 Chaîne de fabrication

8.1 Outils

Application	Lien
Redmine	http://redmine.urssaf.recouv/
Gitlab	http://gitlab.altair.recouv
GLD	http://gld.urssaf.recouv/
Sonarqube	http://sonarqube.urssaf.recouv
Jenkins	http://jenkins.urssaf.recouv
Nexus	http://nexus.urssaf.recouv
Harbor	https://harbor.mutual31.k8s.recouv/
Portail Technique	http://portail.cnp.recouv/portech_portail/fe/#/
FEDEX	http://fedex.cnp.recouv/

8.2 Intégration continue

La chaîne de fabrication est basée sur les outils Git (Gitlab), Jenkins et Nexus.

Les packagings de livraison pour les plateformes HAWAI, SX et pour les bases de données doivent suivre les différentes exigences des différentes chaînes de fabrication ACOSS.

8.2.1 Git

8.2.1.1 Dépôts

Les dépôts des composants devront être créés dans un sous-groupe de l'arborescence suivante :

<http://gitlab.altair.recouv/sded/coeur-de-metier-recouvrement/controle-et-recouvrement-amiable-force-parametres/raf/raf-reнове> .

Le nommage doit suivre une nomenclature `raf-<projet/appli>-<type>-<fonction>`

● #modif Plus de détails sont indiqués dans les exigences "Dépôt - Nomenclature des noms de dépôt" (#16514)

8.2.1.2 Branches

Toutes les sources des composants doivent être poussées dans les dépôts créés pour ce dernier :

Pour la livraison d'un incrément : dans la branche **release/X.Y.Z**

Pour la livraison d'une version : dans la branche **master**

X : version évolutive de production

Y : version incrément de la version

Z : version corrective de qualification SDIT/CNV

La branche master est à l'image de la version livrée à la conformité finale.

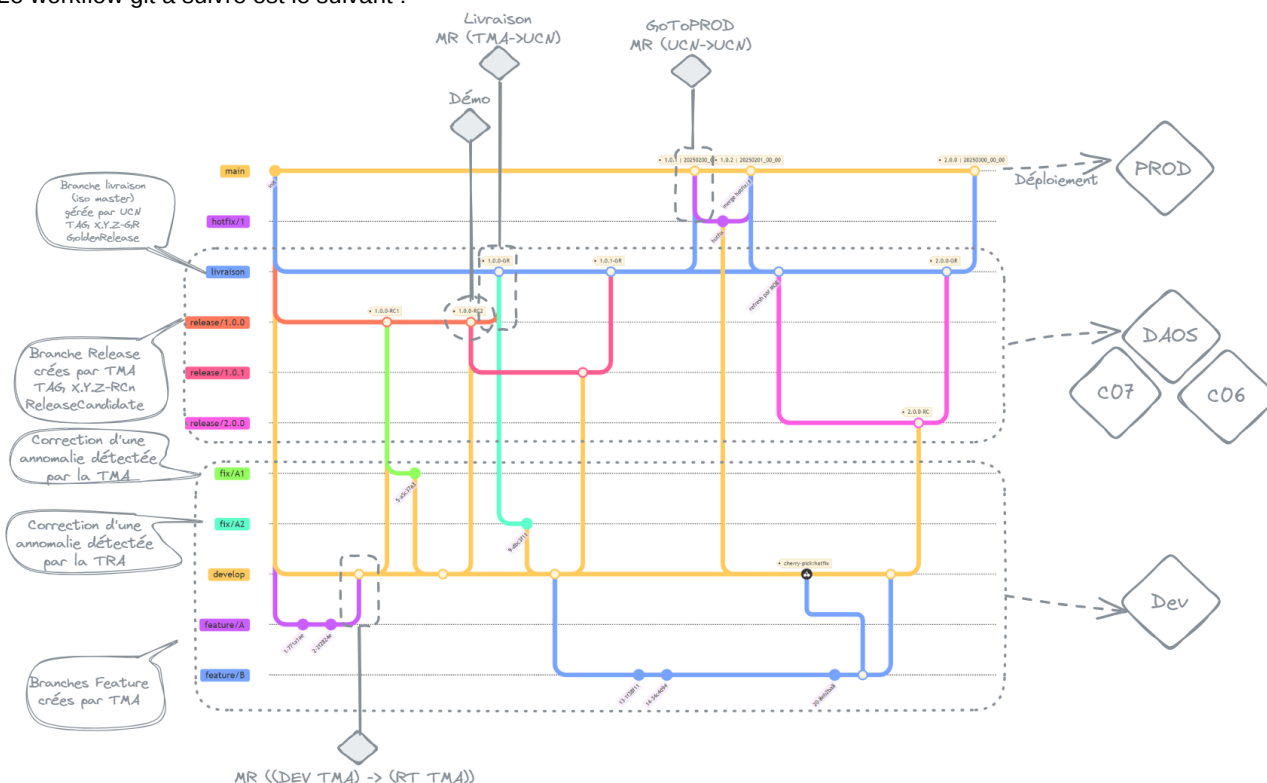
8.2.1.3 Tags

A chaque **livraison d'un incrément**, les sources de chaque composant doivent être taguées avec leur numéro de version au format X.Y.Z-RC (Release Candidate) dans une branche release au format **release/X.Y.Z**.
A la **livraison de la version** ou d'un patch correctif de version, les sources de chaque composant modifiés doivent être taguées avec leur numéro de version sur 3 digits X.Y.Z (Release) dans la branche **master**.

8.2.1.4 Workflow

● #modif #3.3

Le workflow git à suivre est le suivant :



8.2.2 Jenkins

La construction des projets doit être compatible avec le Jenkins de l'UCN et configurable facilement via un JenkinsFile déclaratif basé sur le modèle fourni par le Socle FullStack.

Les actions suivantes doivent pouvoir être pilotées via un ou plusieurs pipelines Jenkins :

- Analyse Sonarqube
- Dependency check
- Trivy
- Tag Git
- Package
- Dépôt sur Nexus
- Création si besoin des paquets Archimed
- Déploiement de la configuration Archimed
- Installation en dev

Organisation des builds doit respecter la hiérarchie du projet dans la DSI :

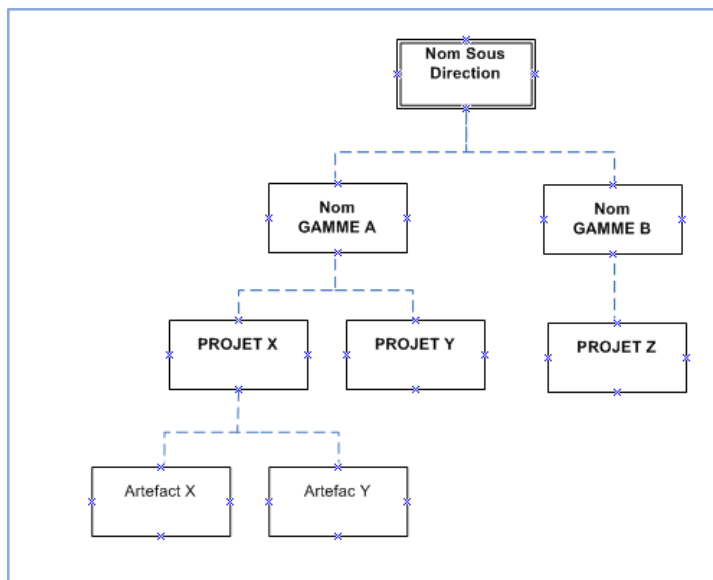
● #modif #3.2

Les binaires développés par l'ESN Titulaire doivent être construits avec la plateforme d'intégration continue ACOSS : NPM / Maven/Jenkins /GIT.

La création et l'exécution des pipelines seront réservées au Release Manager ESN ou responsable d'intégration.

8.2.3 Nexus

L'organisation des artefacts au sein du nexus doit suivre l'organisation suivante :



Les binaires doivent être relâchés et mis à disposition via Nexus avec les caractéristiques suivantes :

- **GroupId** : fr.urssaf.raf.<fonction>.<sous fonction>
- **ArtifactId** : .<nom_module>
- **Incrément** : livraison au format X.Y.Z-RC
- **Version** : livraison au format X.Y.Z

Ex :

<http://nexus.altair.recouv/nexus/content/repository/releases/fr/acoss/recouv/raf/gestionnaire/ihm/raf-gestionnaire-ihm-1.0.1.tar.gz>

Les versions intermédiaires des artefacts dans le cas de build successives (Snapshot, RC) doivent être purgées du serveur nexus par la build Jenkins.

8.2.4 Sonarqube

8.2.4.1 Qualité Java

Les sources développées en Java doivent répondre aux exigences qualité définies par l'ACOSS. Ces exigences sont vérifiées par l'outil Sonarqube via une barrière qualité.

Les règles sont définies dans le fichier suivant dans le profil Java Sonar way dans le [Sonarqube](#) de la nouvelle PIC ACOSS. Ces règles sont mises à jour avec les montées de version de Sonarqube.

Afin de maintenir un niveau de qualité optimale, ces règles doivent être mises à jour régulièrement via l'API Sonarqube. Voir la procédure suivante : <http://gitlab.altair.recouv/bt/outils/pic/-/wikis/Sonarqube/Exporter-Profil-Qualit%C3%A9>

Exemple :

```
curl -u <id_anais>
"http://sonarqube.urssaf.recouv/api/qualityprofiles/export?language=java" -o
"java_code.xml"
```

Le niveau maximum accepté est "majeur". Toutes les mauvaises pratiques, bugs et vulnérabilités de niveau "Critique" et "Bloquant" devront faire l'objet d'une demande auprès des chargés de conformité pour acceptation exceptionnelle.

8.2.4.2 Qualité Typescript

Les sources développées en Typescript doivent répondre aux exigences qualité définies par l'ACOSS. Ces exigences sont vérifiées par l'outil Sonarqube via une barrière qualité.

Les règles sont définies dans le fichier suivant dans le profil Typescript Sonar way dans le [Sonarqube](#) de la nouvelle PIC ACOSS. Ces règles sont mises à jour avec les montées de version de Sonarqube.

Afin de maintenir un niveau de qualité optimale, ces règles doivent être mises à jour régulièrement via l'API Sonarqube. Voir la procédure suivante : <http://gitlab.altair.recouv/bt/outils/pic/-/wikis/Sonarqube/Exporter-Profil-Qualit%C3%A9>

Exemple :

```
curl -u <id_anais>  
"http://sonarqube.urssaf.recouv/api/qualityprofiles/export?language=typescript" -o  
"typescript_code.xml"
```

Le niveau maximum accepté est "majeur". Toutes les mauvaises pratiques, bugs et vulnérabilités de niveau "Critique" et "Bloquant" devront faire l'objet d'une demande auprès des chargés de conformité pour acceptation exceptionnelle.

● #modif #3.2

8.2.5 Dependency Check

L'outil "Dependency Check" de l'OWASP est à mettre en place dans les pipelines Jenkins afin de vérifier les vulnérabilités dans les bibliothèques externes utilisées par le projet.

La procédure à suivre pour intégrer cette étape et rediriger les résultats vers Sonarqube est décrite dans le wiki interne : [How To Dependency Check · Wiki · bt / Outils / PIC · GitLab \(altair.recouv\)](#)

Pour l'analyse des CVE seul le niveau de confiance "Highest" sera pris en compte.

Le niveau maximum des vulnérabilités accepté est "haut". Toutes les vulnérabilités de niveau "Critique" avec un niveau de confiance "Highest" devront faire l'objet d'une demande auprès des chargés de conformité pour acceptation exceptionnelle.

La recherche de vulnérabilités dans les images sera effectuée de manière régulière tout au long de la vie du produit. Les nouvelles failles critiques découvertes devront être prises en compte et corrigées par la partie responsable de l'applicatif au moment de la vérification.

8.3 Livraison

Une livraison consiste en un "Lot Principal" qui peut être constituée de plusieurs lots spécifiques aux socles cibles de l'application.

8.3.1 Lot Refonte

La livraison de la solution doit passer par le déversement du code source dans les dépôts GIT de l'ACOSS et l'exécution d'une build sur la plateforme d'intégration continue interne. Elle est accompagnée par un ensemble de document et de tickets selon son contenu.

8.3.1.1 GIT

Le code source déposé doit comporter suffisamment d'historique de commits GIT pour comprendre les évolutions apportées. Ces derniers devront comporter un commentaire permettant d'identifier les évolutions et correctifs apportés à la solution et inclure les numéros de tickets si existants.

Template : refs #numeroTicketRedmine-- message

Exemple : refs #897254 – Fix label header

● #modif

Les livraisons doivent être effectuées sur des branches "feature" ou "release" et accompagnée d'une merge request vers la branche "develop" ou "master". (exigence : "Livraison - Déversement des sources dans le dépôt URSSAF")

8.3.1.2 Redmine

La livraison d'un lot doit être symbolisée par un ticket redmine racine rassemblant :

- Les tickets redmines spécifiques aux différents socles
- Les documents de livraison
- Les liens vers les MergeRequest sur Gitlab

8.3.1.3 Documents

8.3.1.3.1 STD

● #modif

Les Spécifications techniques détaillées au format asciidoc ● #modif #3.2 doivent être déposées sur le dépôt de documentation sur une branche dédiée. (exigence : "Livraison - Traçabilité des livraisons documentaires DocAsCode").

8.3.1.3.2 Compte Rendu de tests

Les livraisons devront comporter un compte rendu des tests effectués. Ce compte rendu doit aussi montrer que les niveaux de qualités ont été contrôlés et sont sous les seuils attendus ou, dans le cas contraire, justifier les exceptions et fournir mesures de mitigation envisagées.

8.3.1.3.3 Rapports de dépendances

L'ajout de librairie externes au projet doit faire l'objet d'un rapport listant les librairies ajoutées, leurs versions et leurs licences.

8.3.2 Livraison SNV2

La livraison de composant à destination des plateformes SX passe par l'intégration dans un lot SNV2.

8.3.2.1 Encoding

Les fichiers sources doivent être disponibles dans GIT au format ISO-8859-15 pour les développements sous SX (excepté pour les fichiers Liquibase dont l'encoding doit être en UTF-8)

8.3.2.2 GIT

L'intégration des composants dans un lot SNV2 nécessite de commiter les points d'entrées dans le projet SNV2.

Les points d'entrée dans le SNV2 sont les suivants :

Les shells pour les traitements batchs

Les fichiers json et erb pour les containers

L'intégration des modifications du schéma rénové par l'outil liquibase passe également par ce processus.

A partir de la version cible définie dans Redmine, le release manager du fabricant doit créer les branches de référence dans le projet SNV2 de la manière suivante :

Créer un tag PROJET_YYMM00 à partir de la branche de livraison du dépôt 319 /CNTX

Créer une branche DYYMM00 à partir du tag PROJET_YYMM00

Les contributions seront alors commitées sur la branche DYYMM00

8.3.2.2.1 Commit

Le message du commit d'une modification du SI sur la branche DYYMM00 doit comporter la référence du ticket Redmine qu'il corrige.

Ex : Correction du ticket Tracker Anomalie id : 132456

La modification de la demande sera effectuée sur la branche #123456 créée à partir de la branche de la version cible (branche de développement DYYMM00 issue :

Soit de la branche bêta BYYMM00 si lot version projet ,

Soit de la branche livraison si lot mineur,

Soit de la branche release RYYMMXX si correction SDIT ou CNV ou lot hotfix de production)

Le ticket Redmine de chaque demande tracker Anomalie / Evolution / Demande CDS doit comporter le lot dans le champ « Version cible » (exemple : 190100_00_00).

Le message du commit de la correction doit être : « #123456 Correction longueur champ »

Si ce commit corrige un ou plusieurs tickets par exemple #654321 et #234567 alors le message du commit sera le suivant :

« #123456 Correction longueur champ #654321 corrections faute orthographe #234567 corrections lettres accentuées »

8.3.2.3 Batch Nocturne

La livraison d'un batch nocturne dans le SNV2 impose obligatoirement la livraison d'un shell d'appel aux composants Java dans le dépôt Cobol CNTX en respectant très scrupuleusement les normes de développement des batchs régionaux.

Ce shell doit être disponible dans la branche projet correspondant au lot cible.

Ex : si la cible est le lot 1904 alors le fichier doit être présent dans la branche R190400.

Le message du commit de ces fichiers doit contenir l'identifiant du ticket de traçabilité au format #123456

Le dépôt cible est le dépôt Cobol CNTX :

<http://gitlab.altair.recouv/319/cntx>

dans le répertoire shv2.

La nomenclature des fichiers est la suivante:

<code traitement V2>.sh

8.3.2.4 Redmine

8.3.2.4.1 Version

La livraison d'un incrément ou d'une version doit faire l'objet d'un ticket Redmine avec les caractéristiques suivantes :

Tracker : « Livraison »
Sujet : Livraison <Version> - Incrément <n° incrément>
Où Version = [VT1, MEP SOLFI, MEP0, VT2, MEP1, MEP2]
Projet : <Version>
Version cible : <Incrément>
Statut : Livrée Fab (*)
Assigné à : 319 - Release manager
Date de livraison engagée : <date de livraison>
Date de livraison réelle : <date de livraison>
Description

Pré-requis

Description fonctionnelle de la version et de son périmètre

Limitations

Itération MOE Cible : [0-9]* (N° Incrément)

Demande liée à :

Projet SNV2 sur socle SX : tous les trackers « Evolution » référençant les points d'entrée des livrables

Plateforme HAWAI : tous les trackers « Livraison » référençant les livraisons des projets HAWAI

8.3.2.4.2 Patch

La livraison d'un patch correctif d'un incrément ou d'une version doit faire l'objet d'un nouveau ticket Redmine.

Ce ticket doit porter les informations suivantes :

Tracker : « Livraison »
Sujet : Livraison <Version> - Incrément <n° incrément> - patch correctif n° <patch correctif>
Où Version = [VT1, MEP SOLFI, MEP0, VT2, MEP1, MEP2]
Version cible : <Incrément>
Statut : Livrée Fab (*)
Assigné à : 319 - Release manager
Date de livraison engagée : <date de livraison>
Date de livraison réelle : <date de livraison>
Description

Pré-requis

Description fonctionnelle de la version et de son périmètre

Limitations

Itération MOE Cible : [0-9]* (N° Incrément)

Demande liée à :

Projet SNV2 sur socle SX : tous les trackers « Evolution » référençant les points d'entrée des livrables

Plateforme HAWAI : tous les trackers « Livraison » référençant les livraisons des projets HAWAI

8.3.2.4.3 Batch SX

La livraison d'un batch est tracée dans Redmine via un ticket d'évolution ou d'anomalie dans le projet 319/CNTX avec les caractéristiques suivantes :

Projet : SNV2 | 319/CNTX
Tracker : « Evolution »
Sujet : Traitement Batch <code traitement V2>
Version cible : <Version du lot V2 cible>
Statut : Livrée Fab (*)
Assigné à : 319 - Release manager
Date de livraison engagée : <date de livraison>
Date de livraison réelle : <date de livraison>
Code traitement V2 : <code traitement V2>
Code projet : <code projet refonte associé>
Diagnostic Fonctionnel : <Besoin>
Intervention : <Solution>
Description
Descriptif de l'anomalie ou de l'évolution

Demande(s) liée(s) à :
Tracker Livraison type Livrable
Exigence GIT associée

8.3.2.4.4 Conteneur

La livraison des containers sur socle SX doit être tracée dans Redmine via un ticket d'évolution ou d'anomalie dans le projet 319/CNTX avec les caractéristiques suivantes :

Projet : SNV2 | 319/CNTX
Tracker : « Evolution »
Sujet : < code traitement V2> -
Version cible : <Version du lot V2 cible>
Statut : Livrée Fab (*)
Assigné à : 319 - Release manager
Date de livraison engagée : <date de livraison>
Date de livraison réelle : <date de livraison>

Code traitement V2 : <code traitement V2>
Code projet : <code projet refonte associé>
Diagnostic Fonctionnel : <Besoin>
Intervention : <Solution>
Description

Type : Batch nocturne
 Descriptif du traitement/service
 GroupId/ArtifactId/Version
 Batch : classe lanceur
 Qualité : URL du Rapport Sonarqube ACOSS
 Description fonctionnelle
 Demande(s) liée(s) à :
 Tracker Evolution Anomalie du périmètre du conteneur

8.3.2.4.5 Evolution de schéma

La livraison de base de données du schéma RAF rénové sur socle SX est tracée dans Redmine via un ticket d'évolution dans le projet 319/CNTX avec les caractéristiques suivantes :

Projet : SNV2 | 319/CNTX
 Tracker : « Evolution »
 Sujet : Schéma RAF Rénové Version <version schéma>
 Version cible : <Version du lot V2 cible>
 Statut : Livrée Fab (*)
 Assigné à : 319 - Release manager
 Date de livraison engagée : <date de livraison>
 Date de livraison réelle : <date de livraison>
 Code traitement V2 : <code traitement V2 schéma RAF>
 Code projet : <code projet refonte associé>
 Diagnostic Fonctionnel : <Besoin>
 Intervention : <Solution>
 Description
 Schéma impacté : SNV2/RAF
 Version du schéma en entrée :
 Version du schéma en sortie :
 Description des modifications
 Demande(s) liée(s) à :
 Tracker Livraison type Version

8.3.2.4.6 Anomalies SNV2

Chaque livrable produit par l'ESN titulaire émane de demandes d'évolutions ou de corrections d'anomalie effectuées par l'ACOSS.

Chaque ticket Anomalie ou Evolution doit contenir les informations suivantes :

Projet : SNV2 / <Sous-projet>
 Tracker : « Anomalie » ou « Evolution »
 Sujet : <Libellé du ticket>
 Version cible : <Version du composant>
 Statut : Livrée Fab (*)
 Assigné à : 319 - Release manager
 Date de livraison engagée : <date de livraison>
 Date de livraison réelle : <date de livraison>
 Description
 Conteneur
 Conteneur : nom_conteneur / version
 Type : API / Daemon
 API : URI des endpoints et ressources
 Daemon : Topics consommées / publiées
 Description fonctionnelle
 Demande(s) liée(s) à :
 Tracker Livraison type Livrable

8.3.2.4.7 Retour fait par une instance de validation

Lors du retour d'une anomalie provenant de l'étape de transition SDIT/CNV, cette anomalie doit être liée au tracker livraison du patch correctif.

8.3.2.4.8 Redmine pour les conteneurs SX

La création des points d'entrée du Container (Fichier ERB/ Fichier JSON) doit être tracée dans un ticket Evolution au sein du projet SNV2 et suivre le workflow national des livraisons SNV2. Le ticket doit posséder le code traitement V2 défini dans le point 3.2.1. Pour la maîtrise du périmètre, ce ticket doit être lié soit :

- A tous les tickets du périmètre de l'image Docker
- A un tracker livraison de l'image Docker qui est lié à tous les tickets du périmètre

8.3.2.5GLD

Chaque modification de SI est identifiée par un ticket d'anomalie ou d'évolution. Les tickets en visibilité du SDIT doivent faire l'objet d'une fiche de consignes GLD (Guichet de Livraison Documentaire)

La fiche GLD doit être initialisée à partir du ticket Redmine de référence dans l'application :

<http://gld.urssaf.recouv/gld>

Les champs « description » et « intervention » doivent être complétées dans GLD si nécessaire

Les tickets en visibilité du SDIT « corrigés par » doivent également faire l'objet d'une fiche de consigne GLD.

Remarque

Les tickets internes ne font pas l'objet de fiche GLD

Dessins d'enregistrements et fiche de planification

Les documents MODFIC (dessin d'enregistrement) et MODPLA (fiche de planification) doivent être téléversés dans cette dernière dans la section document.

Voir :

-Modèle de dessin d'enregistrement

-Modèle de fiche de planification

GLD pour anomalies

Lors de la livraison d'un ticket Redmine Evolution ou Anomalie, la consigne GLD doit comporter obligatoirement les informations du ticket Redmine comme indiqué dans le chapitre Référence : [Documentation GLD](#) (chapitre 5.1)

GLD pour Doc1

Lors de la livraison d'un nouvel élément doc1, la version du composant s'incrémente. Cette version doit être **obligatoirement** reportée dans la fiche de planification **à téléverser dans la fiche GLD**

Pour les composants DOC1, il faut renseigner le champ « Fichiers adélaide » dans l'onglet « Technique »

Référence : [Documentation GLD](#) (chapitre 6.7. CONSIGNE ADELAIDE)

Consignes spécifiques pour la mise à jour des commandes SNV2

Lors de la création ou mise à jour d'une commande (création ou modification de paramètres PRA1), le développeur doit suivre la procédure de la documentation GLD suivante :

Référence : [Documentation GLD](#) (chapitre 6.11. MISE A JOUR DES COMMANDES)

Consignes pour la création ou la mise à jour d'un nouveau traitement, transaction ou service

Lors de la création ou modification d'un traitement, d'une transaction SAI2 ou d'un service, le développeur doit suivre la procédure de la documentation GLD suivante :

Référence : [Documentation GLD](#) (chapitre 6.7. TRAITEMENTS, TRANSACTIONS ET SERVICES)

Consignes pour la modification de base de données

Lors de la création ou modification de la structure de base de données SNV2, la procédure de la documentation GLD suivante doit être réalisée :

Référence : [Documentation GLD](#) (chapitre 6.5. Table et Fichiers divers)

Informations obligatoires sur la fiche de planification

La fiche de planification doit suivre le template du fichier référencé dans les documents de référence.

L'information « Nombre de ressources » est obligatoire.

Les valeurs possibles sont les suivantes :

01 : pour les commandes globales (traitements)

20 : pour les commandes individuelles (produits individuels et commandes signal)

Informations sur les traitements et transactions impactés

Les modifications des modules peuvent impactés plusieurs traitements Batches ou Transactionnels V2.

En particulier, les modifications des modules « cœur de métier » peuvent avoir des impacts sur de nombreux traitements ou transactions V2.

Dans la consigne GLD liée à la modification de SI, il est obligatoire de renseigner **les traitements et/ou transactions V2 impactés identifiées**.

Les traitements et/ou transactions V2 impactées sont identifiées lors de l'analyse ou le développement de la modification du SI.

Idéalement, Il faut renseigner les impacts réels au niveau le plus fin, c'est à dire utilisant la ou les fonction(s) modifiée(s) dans les sources.

Exemple :

Modification de la fonction F1 du source BDECY.cob

D'un point de vue import dans les modules Cobol, le source BDECY.cob est appelé/utilisé par 51 traitements et/ou transactions V2

Plus finement, la fonction F1 du source BDECY.cob est appelée/utilisée par 3 traitements et/ou transactions V2. Il faut donc indiquer les 3 traitements et/ou transactions V2 dans le champ GLD "Traitements, transactions, services et commandes signals impactés".

Remarque : La consigne GLD comporte deux champs séparés pour renseigner ces informations dans l'onglet Documentation :

- Traitements Impactés
- Transactions impactées

GLD pour la livraison de conteneurs

Actuellement, si de nouvelles variables sont déclarées dans le SX dans le cadre de la livraison sur SX, il faut pousser des consignes dans un fichier au format « libre » pour que l'exploitant prenne en charge ces nouvelles variables sur les environnements dont il a la charge.

Template de **documentation Container Docker** à joindre au GLD : [Template doc Container](#)

La déclaration des variables CASTOR n'est pour le moment pas pris en charge par la documentation technique généré par GLD

Lors de la livraison d'un container, la documentation GLD relative aux consignes CNP sur le container est obligatoire.

Elle doit être jointe dans l'onglet « Documents ».

Modèle de document : **cnp_container_guide_exploitation.doc**

8.3.3 Livraison PFS

8.3.3.1 Nommage des lots

Afin de garantir la traçabilité des livraisons applicatives le n° d'itération doit être modifié à chaque livraison et doit porter de manière sémantique le nombre d'itérations dans l'environnement concerné.

Le format attendu est le suivant : AAMMPP_VV_II avec :

- AAMM: N° du lot sous la forme Année Mois (exemple: 2203 pour Mars 2022)
- PP: N° de l'incrément en production
- VV: N° de l'incrément en validation
- II: N° de l'incrément en intégration
-

Par exemple : le lot 220300_02_01 indique que nous sommes sur la seconde itération d'intégration de la 3ème itération de validation. Le n° d'incrément d'un environnement est remis à 0 lorsque que l'on livre sur l'environnement suivant (en cas de nécessité de relivraison du lot 220300_02_01 en cas de bug découvert en validation on livre une 220300_03_00)

Le but est de pouvoir facilement faire le lien entre livraison technique (tag + chart) et livraison fonctionnelle (Redmine).

8.3.3.2 Redmine

Chaque livraison doit faire l'objet d'un ticket Redmine tracker Livraison, pointant vers le différents livrables.

Si une livraison d'un livrable est refusée pour cause de problème fonctionnel, un nouveau ticket redmine est créé pour la relivraison de la matière. Si le refus est sur les livrables documentaires, la documentation est relivrée sur le ticket existant. Le ticket de Livraison doit suivre le modèle suivant :

Tracker : « Livraison »
Sujet : Livraison <Projet>
Version cible : YYMM00_CC_II
Où YY = année de MEP sur deux chiffres
MM = mois de MEP sur deux chiffres
CC=Itérations CNV sur deux chiffres
II=Itérations SDIT sur deux chiffres

Statut : Livrée Fab ()*
Assigné à : 319 - Release manager
Date de livraison réelle : <date de livraison>
Description
Périmètre : ensemble des composants modifiés :
FrontEnd
Backend
Agent
Base de données

Descriptif du lot
Qualité : URL des Rapports Sonarqube ACOSS
Liens sharepoint :
STD <version livrée>
Matrice des exigences <version livrée>
Version des bundles
Url job jenkins
Url Dépôts gitlab
Sources Nexus
Réserves
Notes

Document de lot

Version Snapshot : <Numero de version>
Demande(s) liée(s) à :
Tous les trackers Anomalie/Evolution liés au périmètre du livrable

8.3.3.3 Documentation

Vérifier la présence et la conformité du document de lot au format readthedoc dans le dépôt documentaire du projet selon le modèle fourni dans DACStack. ● #modif #3.2

Points Importants :

- présence des informations concernant la base de données.
 - Passage d'une migration liquibase
- Si la version de base de donnée ne change pas, NA doit être indiqué.
- le document du Lot précise bien le lieu de stockage du DIT/DDP
 - Vérifier que le lieu renseigné de stockage du DIT/DDP est bien valide, accessible et contient bien le DIT/DDP
- le document du Lot précise bien le lieu de stockage du DEX
- Vérifier que le lieu renseigné de stockage du DEX est bien valide, accessible et contient bien le DEX

-
- Vérification de la présence d'un dossier de spécification fonctionnelle
 - Vérifier dans les tickets d'évol et ano la présence des informations concernant le rapport de tests ESN. Celui-ci doit être disponible au téléchargement.

8.3.4 Livraison Hawai

8.3.4.1 Nommage des lots

Vérifier le nommage du lot

Nomenclature du lot : **YYMM00_CC_II**

- YY = Année sur deux chiffres
- MM = Mois sur deux chiffres
- CC = Itérations CNV sur deux chiffres
- II = Itérations SDIT sur deux chiffres

8.3.4.2 Redmine

Les livrables sur plateforme HAWAI sont des RPMs contenant l'intégralité des composants applicatifs déployés sur une plateforme.

8.3.4.2.1 Version

Chaque livraison doit faire l'objet d'un ticket Redmine tracker Livraison.

Si une livraison d'un livrable est refusée, pour cause de problème fonctionnel, un nouveau ticket redmine est créé pour la relivraison de la matière. Si le refus est sur les livrables documentaires, la documentation est relivrée sur le ticket existant.

Le ticket de Livraison doit suivre le modèle suivant :

```
Tracker : « Livraison »
Sujet : Livraison <Projet HAWAI> - RPM <Nom du RPM versionné>
Version cible : YYMM00_CC_II
Où YY = année de MEP sur deux chiffres
MM = mois de MEP sur deux chiffres
CC=Itérations CNV sur deux chiffres
II=Itérations SDIT sur deux chiffres

Statut : Livrée Fab (*)
Assigné à : 319 - Release manager
Date de livraison réelle : <date de livraison>
Description
Périmètre : ensemble des composants modifiés :
FrontEnd
Backend
Agent
Base de données

Descriptif du lot
Qualité : URL des Rapports Sonarqube ACOSS
Liens sharepoint :
STD <version livrée>
Matrice des exigences <version livrée>

Document de lot

Version Snapshot : <Numero de RPM>
Demande(s) liée(s) à :
Tous les trackers Anomalie/Evolution liés au périmètre du livrable
```

8.3.4.2.2 Anomalie Hawai

Chaque livrable produit par l'ESN titulaire émane de demandes d'évolutions ou de corrections d'anomalie effectuées par l'ACOSS.

Pour les projets HAWAI, ces demandes sont consignées dans un le projet Redmine <à définir>.

Ainsi chaque ticket Anomalie ou Evolution lié à une livraison doit répondre au modèle suivant :

```
Projet : <A définir>
Tracker : « Anomalie » ou « Evolution »
Sujet : <Libellé du ticket>
Version cible : <Version du composant >
Statut : Livrée Fab (*)
Assigné à : 319 - Release manager
Date de livraison engagée : <date de livraison>
Description
Descriptif de l'anomalie ou de l'évolution

Demande(s) liée(s) à :
Tracker Livraison type Livrable
```

8.3.4.3 Encoding

Les fichiers sources doivent être disponibles dans GIT au format UTF-8 pour les développements sous HAWAI

8.3.4.4 Documentation

Vérifier la présence et la conformité du document de lot au format readthedoc dans le dépôt documentaire du projet selon le modèle fourni dans DACStack. ● #modif #3.2

Points Importants :

Si on n'est pas sur la première livraison du train XXY (avec XX pour l'année et YY pour le mois), la documentation de lot doit comporter l'ensemble des informations des livraisons précédentes du train en cours (et notamment doit reprendre l'ensemble des targets de toutes les livraisons précédentes plus celle en cours) ● #modif #3.2

- - présence des informations concernant la base de données.
 - Si update : Num version + target.
 - Si la version de base de donnée ne change pas, NA doit être indiqué.
 - présence des informations sur les propriétés HAWAI applicables. Si pas de nouvelles propriétés à considérer, NA doit être indiqué dans le doc du lot.
 - concordance entre les livrables logiciels listés dans la doc de lot et les composants réellement livrés dans le dépôt SDIT.
 - Le numéro de version du livrable (rpm) ne doit pas dépasser 4 digits et ne doit contenir que des nombres (x.x.x.x max)
 - le document du Lot précise bien le lieu de stockage du DIT/DDP
 - Vérifier que le lieu renseigné de stockage du DIT/DDP est bien valide, accessible et contient bien le DIT/DDP
 - le document du Lot précise bien le lieu de stockage du DEX
 - Vérifier que le lieu renseigné de stockage du DEX est bien valide, accessible et contient bien le DEX
 - Vérification de la présence d'un dossier de spécification fonctionnelle
 - Vérifier dans les tickets d'évol et ano la présence des informations concernant le rapport de tests ESN. Celui-ci doit être disponible au téléchargement.

8.4 Procédures à suivre

8.4.1 *Construction d'application Frontend*

Le développement d'une application frontend doit passer par les étapes suivantes :

- ☐ Construction de maquettes
- ☐ Validation de l'ergonomie
- ☐ Atelier d'intégration PORA
- ☐ Dossier d'intégration PORA
- ☐ Création du groupe et dépôt git principal
- ☐ Ecriture et Livraison des STD
- ☐ Déclaration CADRAN des droits métiers
- ☐ Récupération des identifiants front
- ☐ Mise en place des pipeline Jenkins (application, bundle, déploiement)
- ☐ Développement de l'application
- ☐ Création et installation d'un Bundle Archimed Frontend
- ☐ Ajout des règles nécessaires dans les Bundles Backend utilisés
- ☐ Déploiement de l'application

8.4.2 *Construction d'application Backend*

Le développement d'une application backend doit passer par les étapes suivantes :

- ☐ Création du groupe et dépôt git principal
- ☐ Ecriture et Livraison des STD
- ☐ Identification ou Création du bundle Archimed de publication des APIs
- ☐ Définition des règles de sécurité applicables au APIs
- ☐ Développement de l'application
 - ☐ Projet Java
 - ☐ Projet Chart
 - ☐ Projet Deploy
- ☐ Mise en place des pipeline Jenkins (application, bundle, déploiement)
- ☐ Installation du Bundle Archimed
- ☐ Déploiement de l'application