

Horus

Documentation d'Architecture Technique

Version 1.3 du 09/12/2025

Documentation d'architecture technique

Visas

Rôle	Nom	Date et visa
Auteur	Bastian Charlet	
Valideur Nat System	Déborah Kassabi	
Valideur ABM	David Vitte	

Historique des versions

Version	Date	Auteur	Description
1.0	23/01/2024	Bastian CHARLET	Version initiale du document
1.1	25/11/2025	Bastian CHARLET	Mise à jour
1.2	27/11/2025	Arthur BANCKAERT	Mise à jour

Documentation d'architecture technique

Table des matières

1	Introduction	5
2	Architecture de l'application	6
2.1	Architecture logique.....	6
2.2	Pile technique	6
2.2.1	Backend	6
2.2.2	Frontend	7
2.2.3	Base de Données.....	7
2.3	Structure du projet.....	8
2.4	Poste client	8
2.4.1	Architecture matérielle	9
2.4.2	Poste de développement	11
2.4.3	Service LibreOffice	11
2.4.4	Base de données Oracle SESAME (SA)	11
2.4.5	Base de données Oracle THESAURUS	12
2.4.6	Données issues de CRISTAL	12
2.5	Architecture Angular	14
2.5.1	Vision et objectifs Angular.....	14
2.5.2	Architecture générale	14
2.5.3	Choix technologiques Angular	14
2.5.4	Composants et directives	14
2.5.5	Gestion de l'État	15
2.5.6	Communication avec le Backend	15
2.5.7	Tests	15

Documentation d'architecture technique

2.5.8	Performances.....	15
2.5.9	Outils et bonnes pratiques	15
2.5.10	Gestion des dépendances	15
2.5.11	Évolutions futures.....	15
2.6	Editions.....	15
3	Déploiement et Infrastructure.....	17
3.1	Conteneurisation	17
3.2	Orchestration.....	17
3.3	Configuration	17
4	Annexe.....	18

1 Introduction

L'objet de ce document est de décrire la plate-forme qui supporte l'application Horus.

La présentation de la plate-forme technique se décompose en 2 phases décrivant

- L'architecture logicielle
- L'architecture matérielle

Nous nous plaçons d'un point de vue client, serveur et poste de développement.

2 Architecture de l'application

2.1 Architecture logique

L'architecture mise en place est l'architecture JAVA Spring boot JPA dockerisé. Le schéma suivant présente l'architecture logique d'un point de vue général :

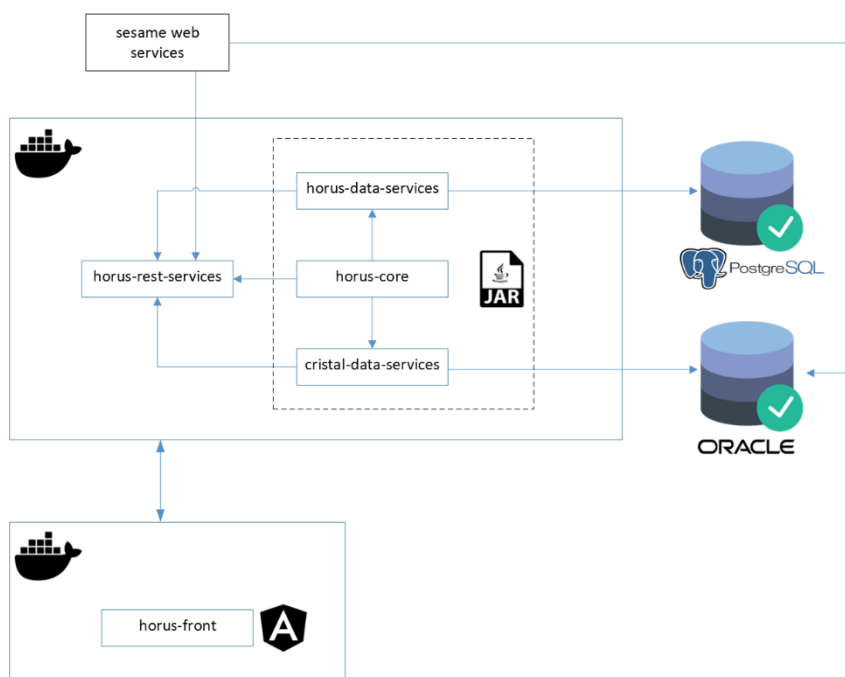


Figure 1 : Architecture Logique

2.2 Pile technique

2.2.1 Backend

Le backend est développé en Java avec le framework Spring Boot.

Technologie	Version	Usage
Java	21	Langage de programmation
Spring Boot	3.5.7	Framework applicatif
Spring Data JPA	(via Spring Boot)	Abstraction ORM
Spring Security	(via Spring Boot)	Sécurité des API
Hibernate	(via Spring Boot)	Implémentation JPA
MapStruct	1.6.3	Mapping Objet-Objet (DTO <-> Entity)
SpringDoc OpenAPI	2.8.14	Documentation API (Swagger)

Documentation d'architecture technique

Log4j2	2.25.2	Journalisation
--------	--------	----------------

- Monitoring : Spring Boot Actuator
- Planification : Quartz Scheduler
- Conversion de Documents : JodConverter
- Utilitaires : Lombok, Commons IO/Net

2.2.2 Frontend

Le frontend est une Single Page Application (SPA) développée avec Angular.

Technologie	Version	Usage
Angular	20.3.15	Framework Frontend
PrimeNG	17.18.10	Bibliothèque de composants UI
Keycloak Angular	16.0.1	Intégration Sécurité
Keycloak-js	25.0.2	Librairie keycloak front
NgRx	20.1.0	Gestion d'état (Store, Effects,...)
RxJS	7.8.1	Programmation réactive
Jest	29.7.0	Tests unitaires
PrimeIcons	7.0.0	Librairie PrimeNG d'icônes
Tailwind CSS	3.4.17	Librairie pour le responsive design
Typescript	5.8.3	

- Qualité de Code : ESLint, Prettier
- Serveur Web : Nginx (pour servir les fichiers statiques)
- Build : npm (intégré via frontend-maven-plugin)

2.2.3 Base de Données

Technologie	Version	Usage
SGBD	19.27	Oracle Database 19c (implied by ojdbc10 version 19.27)

Documentation d'architecture technique

2.3 Structure du projet

Module	Description
horus-parent	POM parent gérant les versions et la configuration globale.
horus-core	Contient le modèle de domaine et la logique métier transverse.
horus-data-services	Couche d'accès aux données principales (Repositories, Entities).
cristal-data-services	Module d'intégration spécifique pour les services de données "Cristal".
horus-rest-services	Couche d'exposition REST (Controllers, DTOs). Point d'entrée de l'application backend.
horus-front	Code source de l'application Angular.
horus-integration	Module indépendant contenant les ressources ainsi que le process de construction du livrable

2.4 Poste client

Sur les postes, un navigateur Internet (*Internet Explorer, Chrome ou Mozilla Firefox*) permet d'utiliser l'application.

Le navigateur accède au serveur qui lui renvoie la page HTML5 dynamiquement générée correspondant à sa demande. De manière générale, à chaque fonctionnalité de l'application correspond une page HTML5 dynamiquement générée différente.

Documentation d'architecture technique

2.4.1 Architecture matérielle

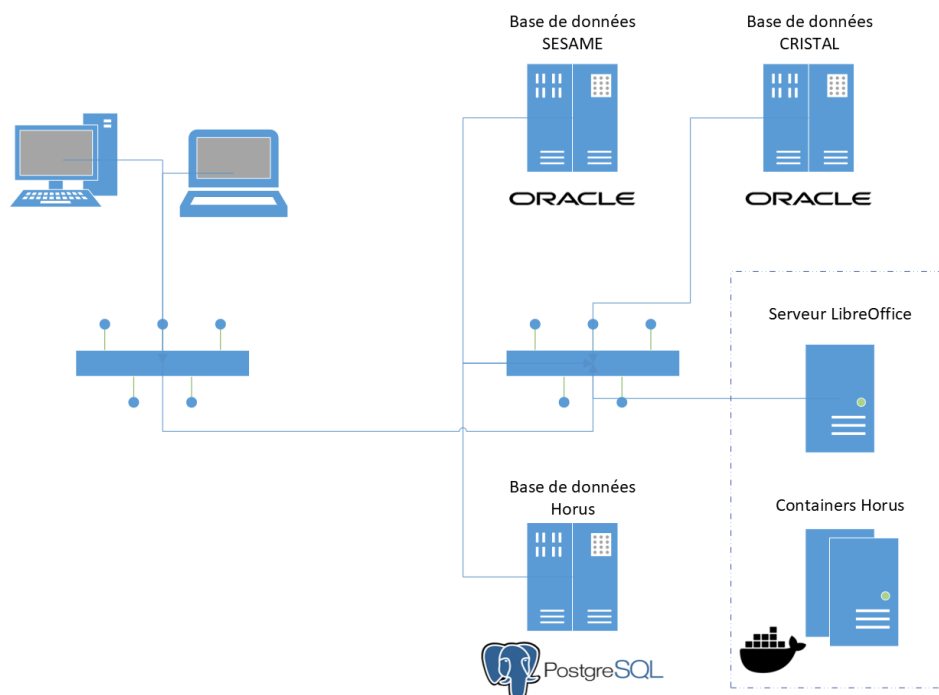


Figure 2 : Architecture matérielle

De	Vers	Flux	Volumétrie	Commentaires
Horus	VGDB	JDBC	N/A	PostgreSQL
Horus	SESAME	Webservices	N/A	Oracle
Horus	THESAURUS	WebServices	N/A	Oracle
Horus	CRISTALRT	JDBC	N/A	Oracle
Horus	Serveur Mail	SMTP	2 mails / message PSQ	Notification d'un nouveau message
			1 mail	Déclaration d'un événement
			0-3 mails	Jusqu'à 3 mails de relance

2.4.1.1 Serveur applicatif et base de données

Configuration :

Composant	Nom	Version
Système d'exploitation	Debian GNU/Linux	11
	Ubuntu	22.04.3 LTS
Serveur d'application	SpringBoot	3.5.7
	httpd	2.4.58

Documentation d'architecture technique

Base de données	Oracle	12C
	PostgreSQL	13.11
JAVA	JDK	21

Sur le serveur sont présents :

- Containers horus-front et horus-rest-services
 - Horus Rest services
 - Aiguille les requêtes provenant du front et appellent les jar concernés.
 - Cristal Data services
 - Traite les requêtes provenant de Horus Rest services et exécute de requêtes vers la base de données CRISTAL
 - Horus Core
 - Couches transverse de classes utilitaires. Contient l'algorithmie récurrente au différent jar. Ce jar est commun à tous les projets qui composent l'application Horus
 - Horus Data services
 - Traite les requêtes provenant de Horus Rest services et exécute de requêtes vers la base de données VGDB

Ces points particuliers peuvent être mis en œuvre avec 2 approches de développement :

Documentation d'architecture technique

2.4.2 Poste de développement

Composant	Nom	Version
Eclipse		4.19.0
Intel IJ		latest
JAVA	JDK	21
Bibliothèque des éditions	Birt	4.13.0
Client Base de données	SQL Developer	22.2.1.234
	PgAdmin4	4.0
Serveur d'application	Jetty	11.0.26
	Apache httpd	2.4.58

2.4.3 Service LibreOffice

Composant	Nom	Version
Système d'exploitation	RedHat	CentOS7
Service d'édition	LibreOffice	5.3.6.1

Le service LibreOffice permet d'effectuer les conversions des fichiers vers un format PDF.

Le paramétrage afférent à l'accès au service est configuré dans le fichier application.properties.

2.4.4 Base de données Oracle SESAME (SA)

L'application n'a de data source dédiée vers la base de données SESAME. Ces tables sont accessibles via des webservices mise à disposition par l'agence de la Biomédecine.

<https://<host>:<port>/SesameWS/swagger-ui.html#>

Les différents services utilisés par Horus sont :

- /acteur-coordonnees
- /acteurs
- /départements
- /entites
- /etablissements
- /fonctions
- /pays
- /reseaux

Documentation d'architecture technique

2.4.5 Base de données Oracle THESAURUS

L'application n'a de data source dédiée vers la base de données THESAURUS. Ces tables sont accessibles via des webservice mise à disposition par l'agence de la Biomédecine.

<https://<host>:<port>/ThesaurusWS/swagger-ui.html#>

Les différents services utilisés par Horus sont :

- [/concepts](#)
- [/concepts/arbre/simple](#)
- [/nomenclature/libelleNomenclature](#)

2.4.6 Données issues de CRISTAL

L'application n'a pas d'écran dédié à son administration propre. Elle récupère les informations liées aux dons/greffes du référentiel CRISTAL. L'application requête sur la vue VDVR du schéma CRISTAL restreint : CRISTALRT. Le schéma CRISTALRT s'appuie également sur le schéma GREEN.

Les champs utilisés sont les suivants :

Nom du champ	Type	Description
DATE_ETAT_GREFFON	DATE	Date de l'état du greffon
DDON_COD	NUMBER(8)	N° Cristal (donneur)
DDON_DNAI	DATE	Date de naissance du donneur
DDON_PCMC_COD	VARCHAR2(8)	Site de décès et de prélèvement
DDON_PCMC_LIB	VARCHAR2(83)	Libellé du site de décès
DDON_PRL	VARCHAR2(1)	Prélevé organe ? (O ou N)
DDON_PRT	VARCHAR2(1)	Prélevé tissu ? (O ou N)
DDON_SEX	VARCHAR2(1)	Sexe du donneur
DDON_PCMC_ETA_CODE	VARCHAR2(13)	Code Etablissement du site de décès et de prélèvement
DORG_PDTO_POG	VARCHAR2(4)	Partie d'organe
DORG_PORG_COD	VARCHAR2(8)	Organe prélevé
DORG_PDTO_LIB	VARCHAR2(60)	Libellé complet de l'organe prélevé
DORG_PORG_TYP	VARCHAR2(1)	Type d'organe (O = organe, T=tissus, X=tissus composite)

Documentation d'architecture technique

DORG_PRELEVE	VARCHAR2(1)	Organe prélevé (O = Oui / N = Non / null = Pas d'info)
DORG_GREFFE	VARCHAR2(1)	Organe greffé (O = Oui / N = Non / null = Pas d'info)
EQUIPE_GREFFON	VARCHAR2(40)	Selon l'état, équipe d'attribution si attribué, équipe destinatrice si prélevé
EQUIPE_GREFFON_LIB	VARCHAR2(40)	Libellé de l'équipe de greffe
ETAT_GREFFON	VARCHAR2(37)	Etat du greffon (Attribué, prélevé, greffé, information indisponible)
PTYD_COD	VARCHAR2(8)	Type de donneur (SME, DDAC, etc...)
PTYD_ETD	VARCHAR2(1)	Type de donneur (D pour décédé, V pour vivant)
RORG_COD	NUMBER(8)	NATT (receveur)
RORG_DGRF	DATE	Date de greffe
RORG_PCMC_COD	VARCHAR2(8)	Etablissement d'attente ou de greffe
RORG_PCMC_LIB	VARCHAR2(83)	Nom de l'établissement d'attente ou de greffe
RORG_PCMC_SA_REF	NUMBER	ID de l'établissement
RORG_PORG_COD	VARCHAR2(8)	Organe attendu
RORG_PCMC_ETA_CODE	VARCHAR2(13)	Code Etablissement du site de greffe au receveur
RORG_PEMC_COD	VARCHAR2(8)	Equipe d'attente ou de greffe
RREC_COD	NUMBER(8)	NEFG (Numéro Etablissement Français de la Greffe)
RREC_DNAI	DATE	Date de naissance du receveur
RREC_SEX	VARCHAR2(1)	Sexe du receveur

2.5 Architecture Angular

2.5.1 Vision et objectifs Angular

2.5.1.1 Objectifs Angular

- Adopter un *framework* robuste pour le développement côté client (front).
- Assurer la maintenabilité et la scalabilité du code Angular.
- Offrir une expérience utilisateur optimale grâce à des interactions fluides et réactives.

2.5.1.2 Bénéfices attendus

- Développement plus rapide avec des fonctionnalités modulaires.
- Facilité de maintenance grâce à la séparation des responsabilités.

2.5.2 Architecture générale

2.5.2.1 Structure du projet

Le projet est structuré selon le modèle de modules Angular, avec une division claire entre les fonctionnalités. Chaque module encapsule des composants, services et directives liés.

Le chargement de chaque module est fait grâce à du *lazy-loading*, ce qui permet de ne charger que les dépendances nécessaires.

2.5.2.2 Modules

- **CoreModule** : contient les services nécessaires à tout moment (authentification, mise en page, *interceptors*, *guards*, configuration)
- **SharedModule** : contient les services partagés par plusieurs parties indépendantes
- **FeatureModule** : contient les services propres à chaque fonctionnalité.

2.5.3 Choix technologiques Angular

- **Angular 20** : adopte la dernière version stable d'Angular (lors du début du projet) afin de bénéficier des dernière fonctionnalités et corrections.
- **Typescript** : Utilisation de Typescript pour tirer parti du typage statique

2.5.4 Composants et directives

- Les composants sont utilisés de manière extensive, avec une attention particulière à la réutilisation.
- Directives personnalisées pour les fonctionnalités spécifiques à notre application.

Documentation d'architecture technique

2.5.5 Gestion de l'État

- Utilisation de *NgRX* pour la gestion de l'état global (store). Cela permet de garder en mémoire, dans le navigateur, des données qui sont amenées à être rechargées fréquemment.
- Utilisation des services Angular pour l'état local des composants.

2.5.6 Communication avec le Backend

- Ce sont les services Angular qui envoient les requêtes (http) à l'API REST pour les opérations CRUD.
- Utilisation des *Interceptor* Angular pour la gestion globale des headers (*X_CODE-SESAME*, *X_CODE-CENTRE*, *X-EN-TANT-QUE*, *X-PROFIL-HORUS*) ainsi que la gestion des erreurs 401 et 403.

2.5.7 Tests

- Stratégie de tests unitaires pour les services et composants critiques avec la librairie *Jest*.

2.5.8 Performances

- Optimisation du chargement initial avec le pré-chargement sélectif des modules.
- Gestion fine des états de chargement lors des appels API.

2.5.9 Outils et bonnes pratiques

- Angular CLI pour le développement, le test et le déploiement.
- Respect des conventions de codage Angular (Angular.io)

2.5.10 Gestion des dépendances

- Gestionnaire de paquets *npm* pour les dépendances tierces.

2.5.11 Évolutions futures

- Le développement de l'application a été réalisé de manière à faciliter la migration vers des versions Angular ultérieures.

2.6 Editions

Cette fonctionnalité est réalisée par la librairie suivante :

[org.eclipse.birt.runtime-4.4.2.jar](#)

Documentation d'architecture technique

Le moteur BIRT récupère les adresses et identifiants des bases Cristal et Horus, ainsi que l'identifiant de la déclaration et crée l'édition à partir des fichiers **.rptdesign**.

Ces fichiers contiennent les requêtes SQL nécessaires à l'alimentation des champs de l'édition.

Il existe 2 éditions :

- **declaration.rptdesign** : qui reprend l'ensemble des rubriques de la déclaration ainsi que la requalification, les échanges et les avis d'experts liés à la déclaration
- **expertise.rptdesign** : qui reprend le détail d'un avis d'experts

3 Déploiement et Infrastructure

3.1 Conteneurisation

L'application est conteneurisée utilisant **Docker**.

- **Backend** : Image construite via jib-maven-plugin ou Dockerfile.
- **Frontend** : Image construite via Dockerfile, utilisant Nginx pour servir les assets compilés.

3.2 Orchestration

Un fichier

docker-compose.yml est fourni pour l'orchestration locale ou simple.

- **Service backend** : Expose le port 8080. Monte les fichiers de configuration (application.properties, logs, etc.) via des volumes.
- **Service frontend** : Expose le port 4200 (mappé sur le 80 du conteneur). Dépend du service backend.
- **Réseau** : Les services communiquent via un réseau dédié horus-network.

3.3 Configuration

La configuration est externalisée (Externalized Configuration) pour respecter les principes du *Twelve-Factor App*. Les fichiers de propriétés sont injectés au démarrage du conteneur :

- application-horus.properties
- application-horus-job.properties
- application-horus-swagger.properties
- logback-spring.xml (Configuration des logs)

4 Annexe

Figure 1 : Architecture Logique	6
Figure 2 : Architecture matérielle.....	9
Figure 3: MPD	19
Figure 4 : MDP simplifié	20

Documentation d'architecture technique

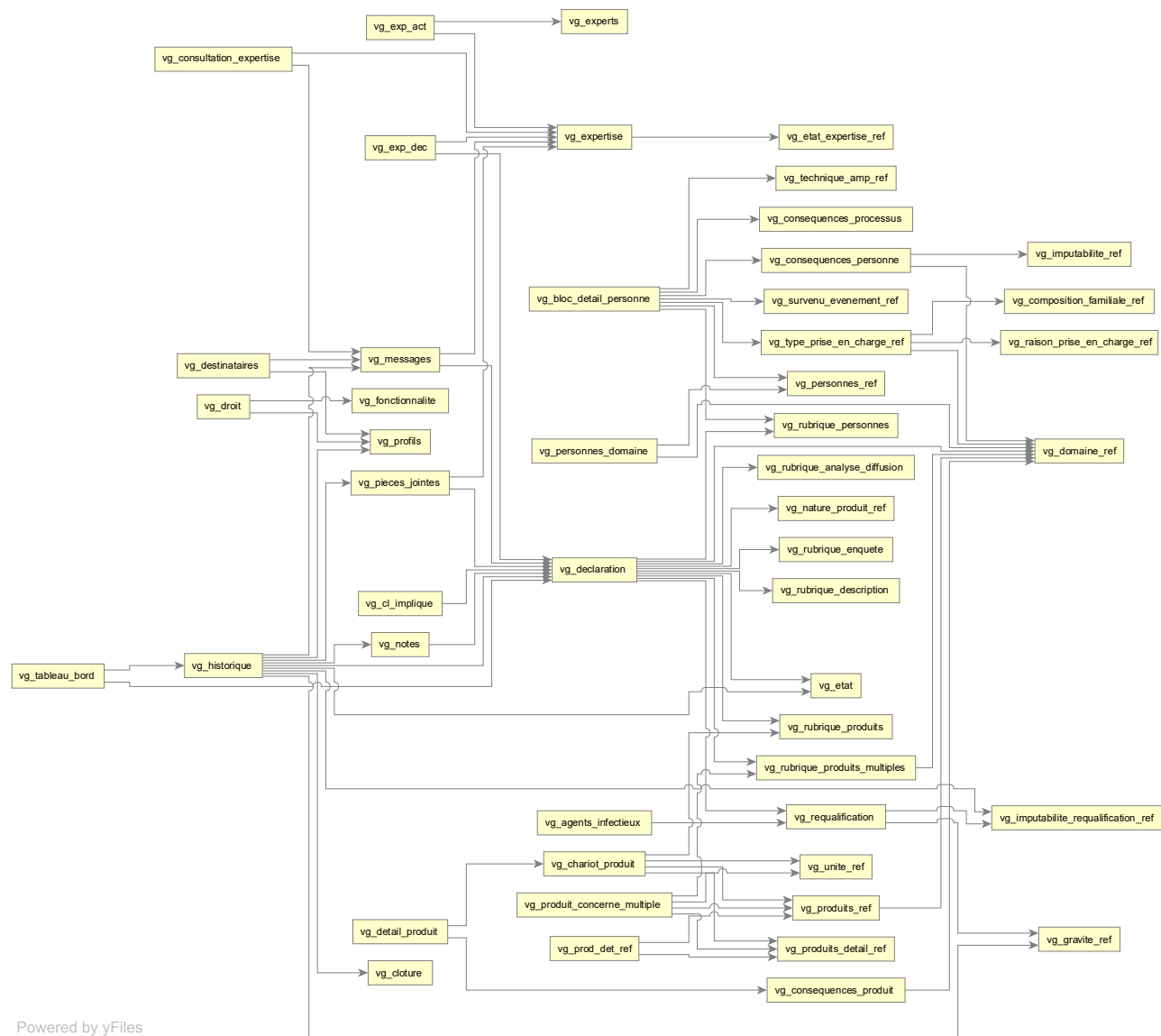


Figure 4 : MDP simplifié