



Ecole Française d'Athènes – Service informatique
6 rue Didotou, 106 80 Athènes, Greece.
Téléphone : +30 210 36 79 996

– Porphyre 2003 –

Serveur de Structure

- Dossier Général -

Version 2.0

Date de création : 29/08/2003

Dernière modification : 29/08/2003

État du document : Terminé

Nombre de pages : 35

Rédacteurs : Baptiste Meurant

Julien Gossa

Guillaume Deshors

SOMMAIRE

1.	INTRODUCTION.....	4
1.1.	PRESENTATION DU PROJET	4
1.2.	OBJET DU DOCUMENT	4
1.3.	TERMINOLOGIE.....	4
2.	DEVELOPPEMENT	5
2.1.	DESCRIPTION	5
2.2.	LES DIFFERENTES FONCTIONNALITES.....	5
2.3.	SCHEMAS D'INTEGRATION DU SERVEUR DE STRUCTURE DANS L'ARCHITECTURE MULTI-TIERS DE PORPHYRE	6
2.3.1.	Structure DataBase	8
2.3.1.1.	<i>Schéma de conception</i>	<i>8</i>
2.3.1.2.	<i>Description des objets, interactions</i>	<i>9</i>
2.3.1.3.	<i>Implémentation</i>	<i>13</i>
2.3.1.4.	<i>PorphyryStructureProtocol.....</i>	<i>16</i>
2.3.2.	Class StructureServer	33
3.	SYNTHESE.....	35

1. Introduction

1.1. Présentation du projet

Développer le serveur de structure du projet porphyre 2003 représentant la couche la plus haute dans la « hiérarchie » des serveurs et constituant le principal interlocuteur du client natif.

1.2. Objet du document

Ce document a pour objectif, dans un premier temps, la définition, pour ce serveur de structure, des besoins et des attentes en matière de services.

Dans un second temps, ce document a été mis à jour pour respecter les modifications effectuées et garder une trace de l'ensemble du travail effectué sur ce serveur. Dans cette optique, des informations concernant l'implémentation, notamment celle de la base de donnée ont été ajoutée, ce qui fait de ce document, non plus un document de spécification, telle qu'était sa fonction initiale mais bien un document général concernant le serveur de Structure.

1.3. Terminologie

Contexte de lecture : ensemble d'objets documentaires sémantiquement liés.

Descriptions de corpus : elles se font à l'aide de graphes acycliques de descripteurs et sont mises en contexte suivant les relations de généralisation/spécialisation.

Étape de lecture : elle est définie, dans un parcours de lecture, comme un objet documentaire parmi un corpus et est mise en contexte par les relations de séquence indiquées dans ce parcours de lecture.

Identifiant : (Identifiant) ; il s'agit d'un descripteur (nœud du graphe) particulier. Il représente les documents qui sont à l'intérieur d'un répertoire.

Objet Documentaire : généralisation des notions de source, fragment et note ou d'un ensemble d'autres objets documentaires. Il est caractérisé par sa localisation.

Parcours de lecture (Reading Trail) : suite d'étape de lectures créée par un utilisateur. Ils constituent des « raccourcis » entre des corpus de documents, transversalement aux relations d'inclusion.

Serveur de Contenu : stocke, diffuse et traite les documents sources et les corpus.

Serveur de Correspondance : construit l'URL d'obtention d'un objet documentaire à partir de l'identifiant donné par le serveur de structure.

Trace : peut être un objet documentaire, un corpus de documents, une description semi-formelle d'un corpus ou encore une étape de lecture.

2. Développement

2.1. Description

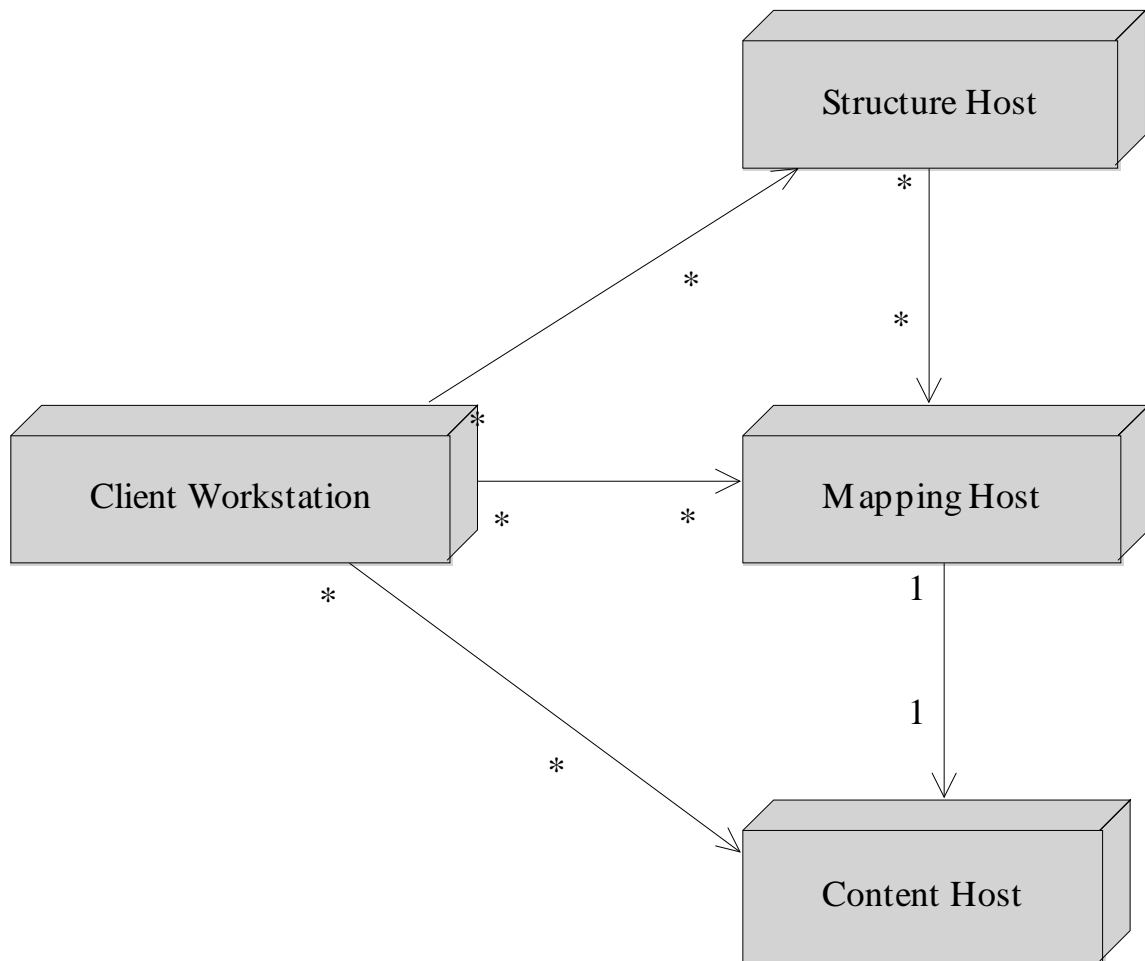
Le serveur de structure doit permettre de stocker les traces et de les représenter dans un contexte susceptible d'intéresser le lecteur. De manière générale, ce serveur – via sa base de données propre – doit permettre le stockage et la restitution des arbres de recherches et de lectures des objets documentaires présents dans le serveur de contenu et qui lui seront accessibles via le serveur de correspondance.

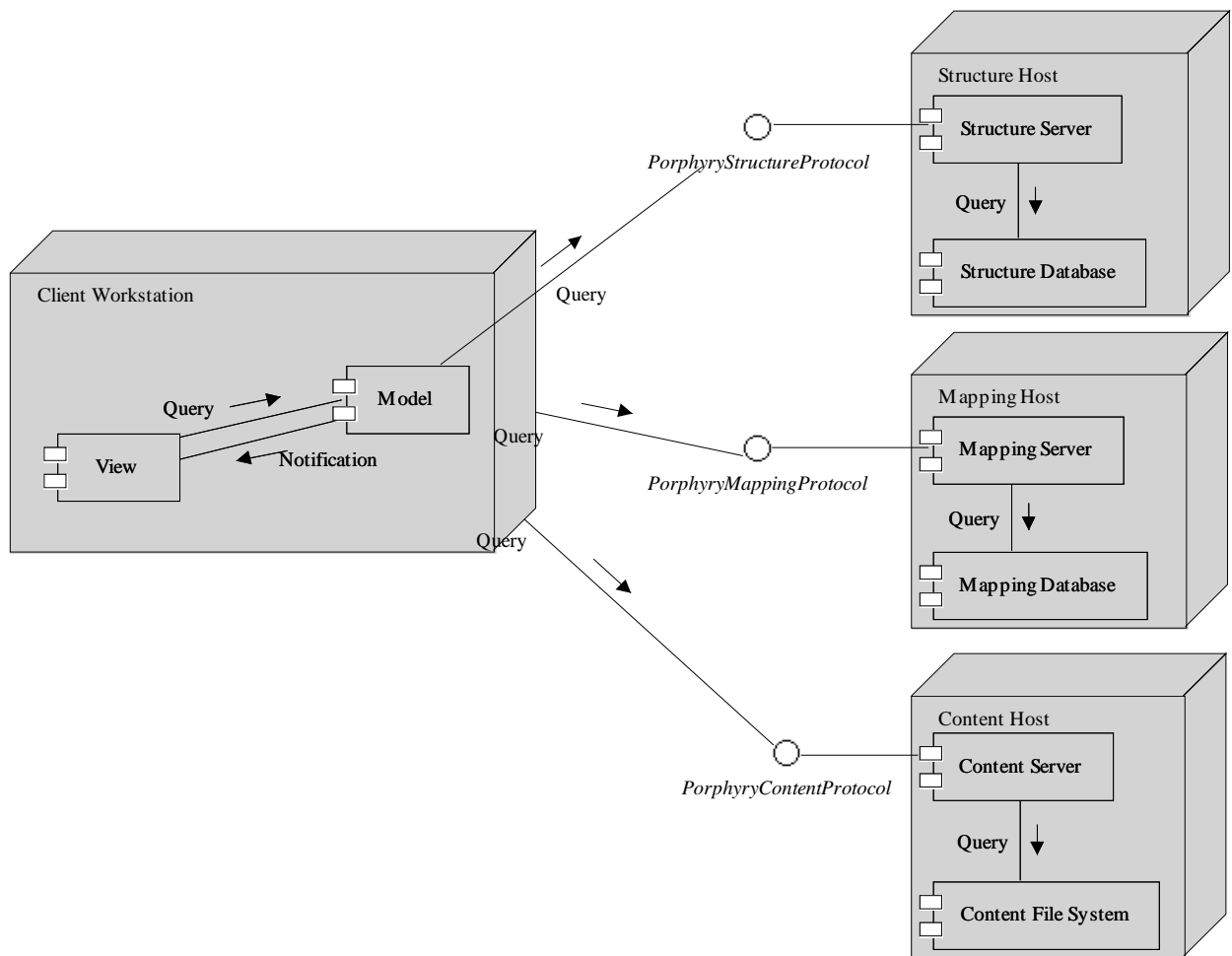
Ce serveur est un point clef de la nouvelle architecture de Porphyre puisqu'il est en charge, outre du stockage des données via sa base de données dédiée, de l'ensemble des calculs et des traitements effectués sur ces mêmes données.

2.2. Les différentes Fonctionnalités

- Stocker l'ensemble des informations nécessaires à la construction (graphique) des réseaux de description et des parcours de lecture par l'application cliente.
- Mettre à disposition de l'application cliente le protocole permettant d'accéder et d'interpréter la base de donnée.
- Éviter toute redondance, incohérence et obsolescence des données en stockant toutes les informations dans une unique base de donnée.
- Garantir la sûreté de la base de données en mettant en place un protocole d'identification des utilisateurs.
- Mettre à disposition de l'application cliente un ensemble de services permettant l'ajout, la suppression, la modification de graphes et de documents de manière transparente et sécurisée.
- Mettre à disposition de l'application cliente un ensemble de services permettant la consultation d'objets documentaires et la navigation à l'intérieur des graphes et des parcours de lecture.
- Effectuer, sur les données stockées, les calculs et les traitements nécessaires aux autres modules de l'application (*model, view*)
- Proposer des outils d'administration du système pour les utilisateurs autorisés (ajout d'utilisateur, etc.).
- Déterminer le serveur de correspondance et l'identifiant permettant d'accéder à l'objet documentaire demandé par le client sous la forme d'un descripteur.

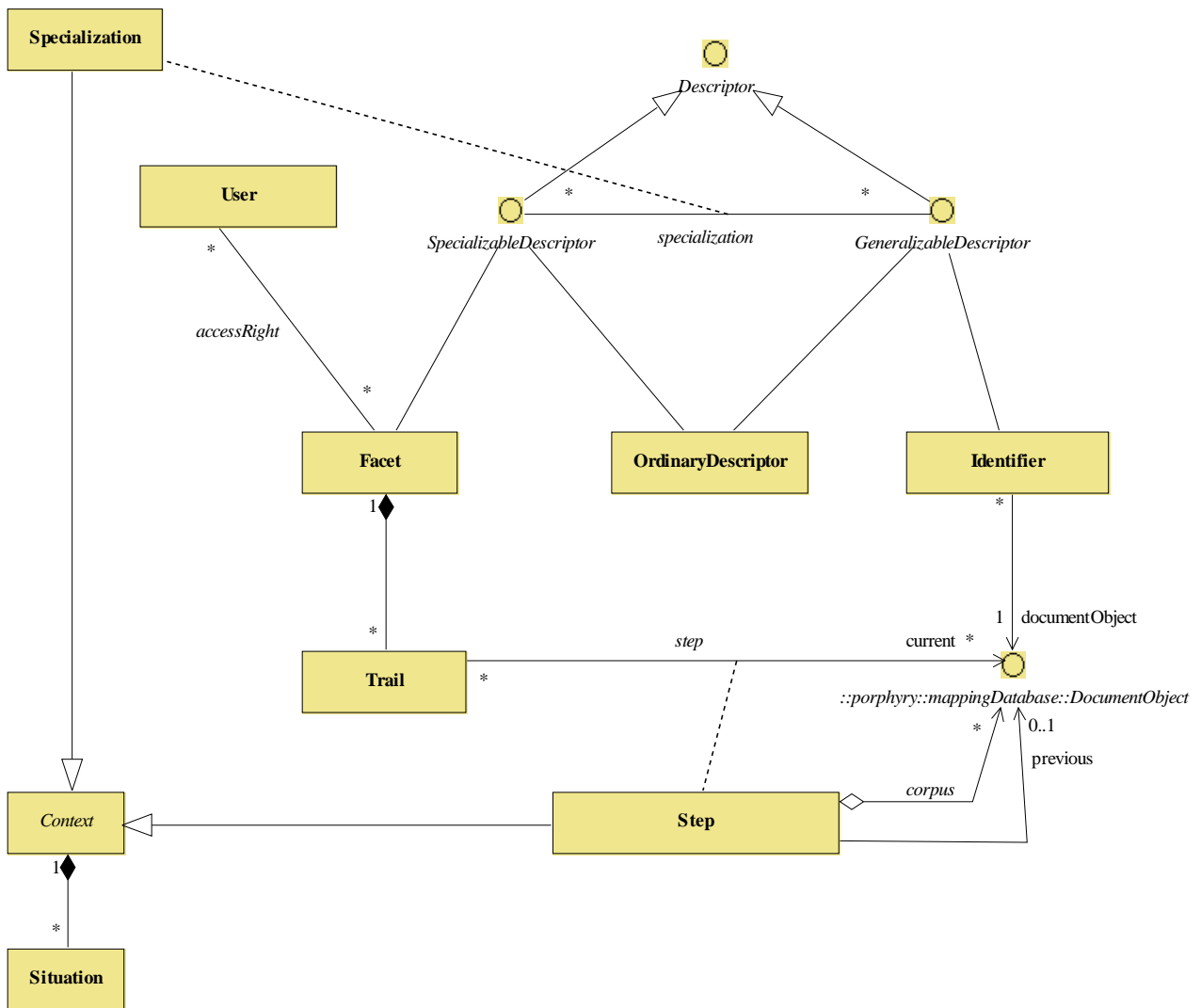
2.3. Schémas d'intégration du serveur de structure dans l'architecture multi-tiers de Porphyre





2.3.1. Structure DataBase

2.3.1.1. Schéma de conception



2.3.1.2. Description des objets, interactions

1. Descriptor

Un *Descriptor* modélise un nœud du graphe de description au sens large. La nature exacte de ce nœud dépend de son caractère généralisable et/ou spécialisable.

2. SpecializableDescriptor

Un *SpecializableDescriptor* est un *Descriptor* qui ne représente pas un élément terminal (feuille) du graphe.

3. GeneralizableDescriptor

Un *GeneralizableDescriptor* est un *Descriptor* qui ne représente pas la racine du graphe.

Un *Descriptor* constitue ainsi la racine du graphe s'il est seulement spécialisable (cf. *Facet*), un nœud au sens classique s'il est à la fois généralisable et spécialisable (cf. *OrdinaryDescriptor*), un élément terminal de ce graphe s'il n'est que généralisable (cf. *Identifier*)

4. Facet

Un objet de type *Facet* correspond à la racine du graphe de description courant ; il est modélisé par un *Descriptor* non généralisable (uniquement spécialisable). Une facette est associée à un certain nombre d'utilisateurs qui disposent de droits plus ou moins importants sur les objets qu'elle décrit (cf. *User* et *AccesRight*). Il est possible, à partir de l'identifiant d'une facette, de retrouver l'ensemble des parcours de lecture auxquels elle est associée (cf. *Trail*).

5. OrdinaryDescriptor

Un objet de type *OrdinaryDescriptor* correspond à un nœud du graphe au sens usuel ; c'est à dire à un objet qui ne constitue ni la racine ni une feuille du graphe. Il est modélisé par un *Descriptor* à la fois spécialisable et généralisable et a donc un père et un ou plusieurs fils.

6. Identifier

Un objet de type *Identifier* correspond à une feuille du graphe courant. Il est modélisé par un *Descriptor* uniquement généralisable (possède un père). Un *Identifier* est associé à un document object (nom et emplacement physique).

7. User

Un objet de type *User* correspond à un utilisateur enregistré du système Porphyry. Il est doté d'un login et d'un password tous les deux constitués de 8 caractères fermes. Une fois enregistré, cet utilisateur dispose, par ce compte, de droits d'accès (cf. *AccessRight*) plus ou moins importants sur un ensemble de facettes.

8. AccessRight

La relation *AccessRight* définit les droits d'accès entre un utilisateur enregistré (*User*) et un graphe (*Facet*). Le premier peut ainsi disposer de droits d'écriture sur cette facette s'il en est le créateur ou de droits de lecture seule dans le cas contraire.

9. Context

Un Contexte permet d'associer des objets documentaires entre eux dans un réseau de description ou un parcours de lecture.

Il s'agit d'une table abstraite et le type effectif de cet objet (*Specialization* ou *Step*) dépend de notre désir d'intégrer ce document dans le réseau de description ou le parcours de lecture.

Le contexte permet également de construire un historique des traitements effectués sur ces documents et leurs liens. Les attributs privés ou publics permettent de distinguer les espaces du lecteur et du scripteur et ainsi de donner des droits spécifiques sur l'objet.

10. Specialization

Les *Specialization* permettent de construire l'arbre entier en construisant la hiérarchie des descripteurs, créant un lien entre un père (*SpecializableDescriptor* : *Facet* ou *OrdinaryDescriptor*) et un fils (*GeneralizableDescriptor* : *OrdinaryDescriptor* ou *Identifier*). Une *Specialization* est un contexte de lecture (*Context*) et on accède ainsi à son historique.

11. Situation

Une *Situation* décrit toute action réalisée à un certain moment par une certaine personne sur l'arbre. La personne en question peut ne pas être un utilisateur enregistré. Une action peut être une Création, une Publication ou une Suppression. Une *Situation* est associée à un objet *Context* ce qui permet, pour un *Context* donné, d'en reconstituer l'historique.

12. Trail

Un objet de type *Trail* modélise un parcours de lecture pour une facette donnée. Un parcours de lecture est constitué d'étapes de lectures (cf. *Step*) chaînées entre elles. Ainsi, on donne un sens à la façon et à l'ordre dans lequel les documents ont été consultés et plus seulement à leur relation de spécialisation.

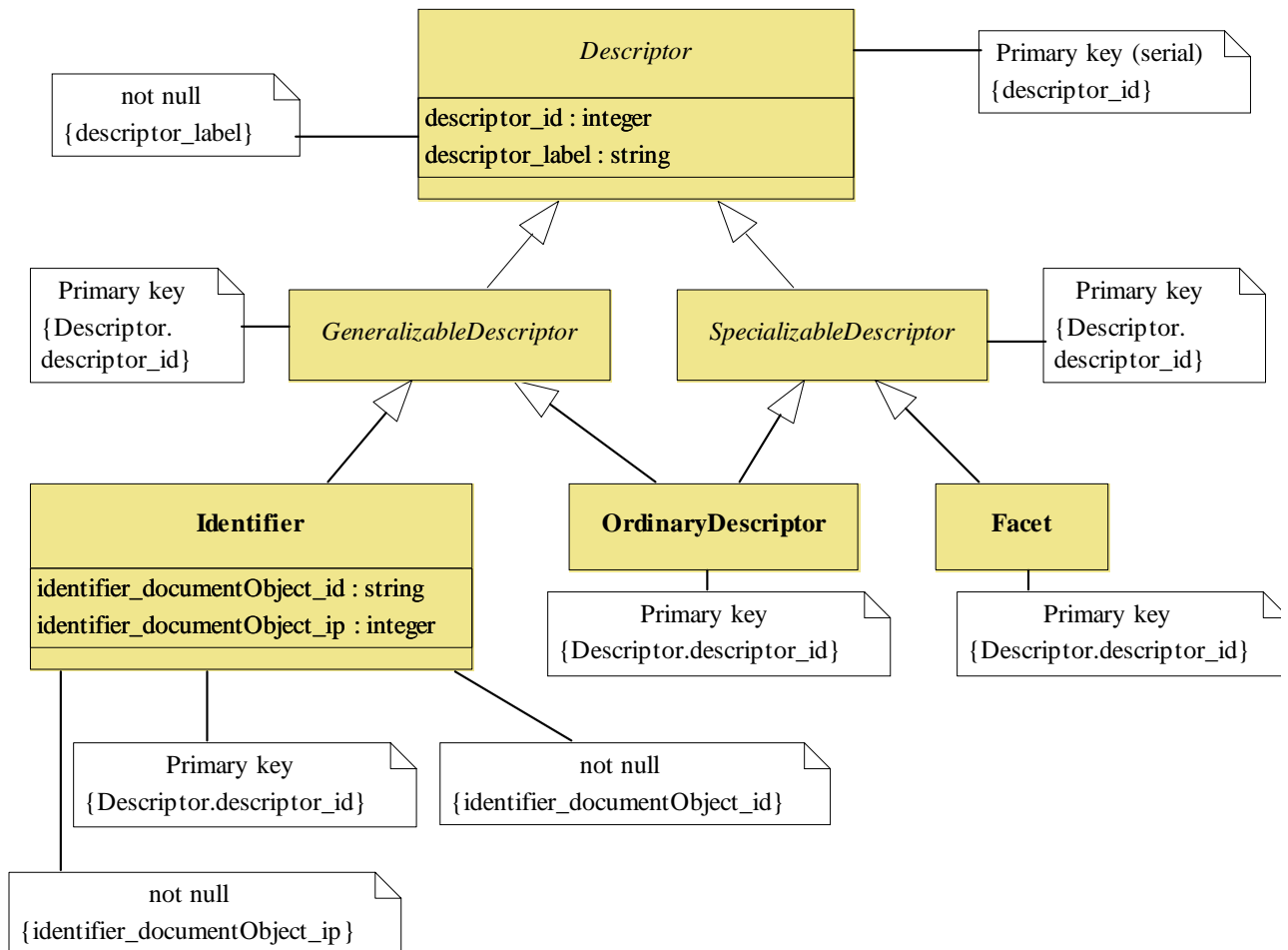
13. Step

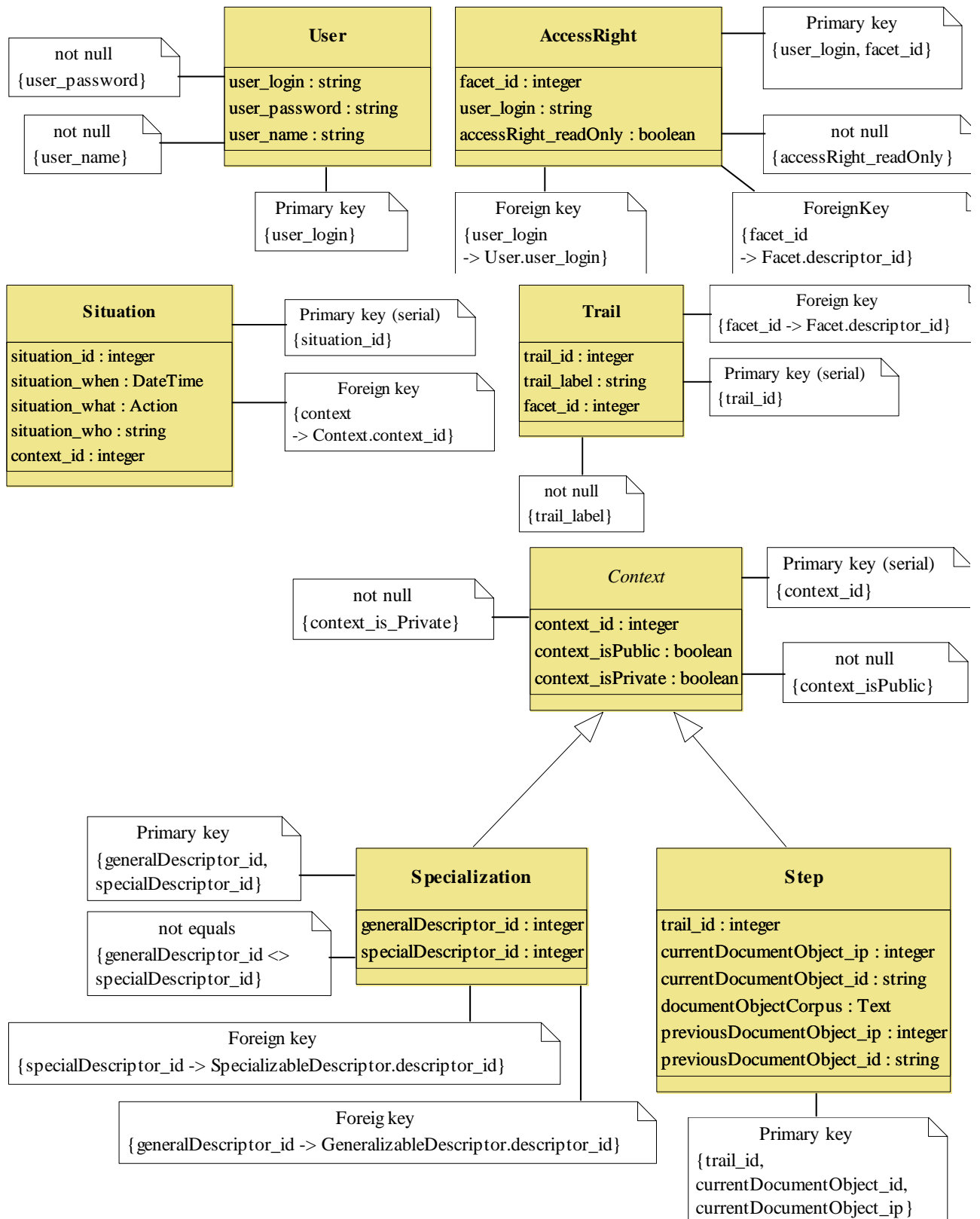
Cet objet modélise une étape de lecture et définit, de proche en proche, le parcours de lecture dont il fait partie (*Trail*). Pour cela, une étape donnée connaît son unique prédécesseur ainsi que le propre objet documentaire auquel elle fait référence. Chaque étape de lecture est un contexte de lecture (*Context*), permettant d'accéder à son historique.

2.3.1.3. Implémentation

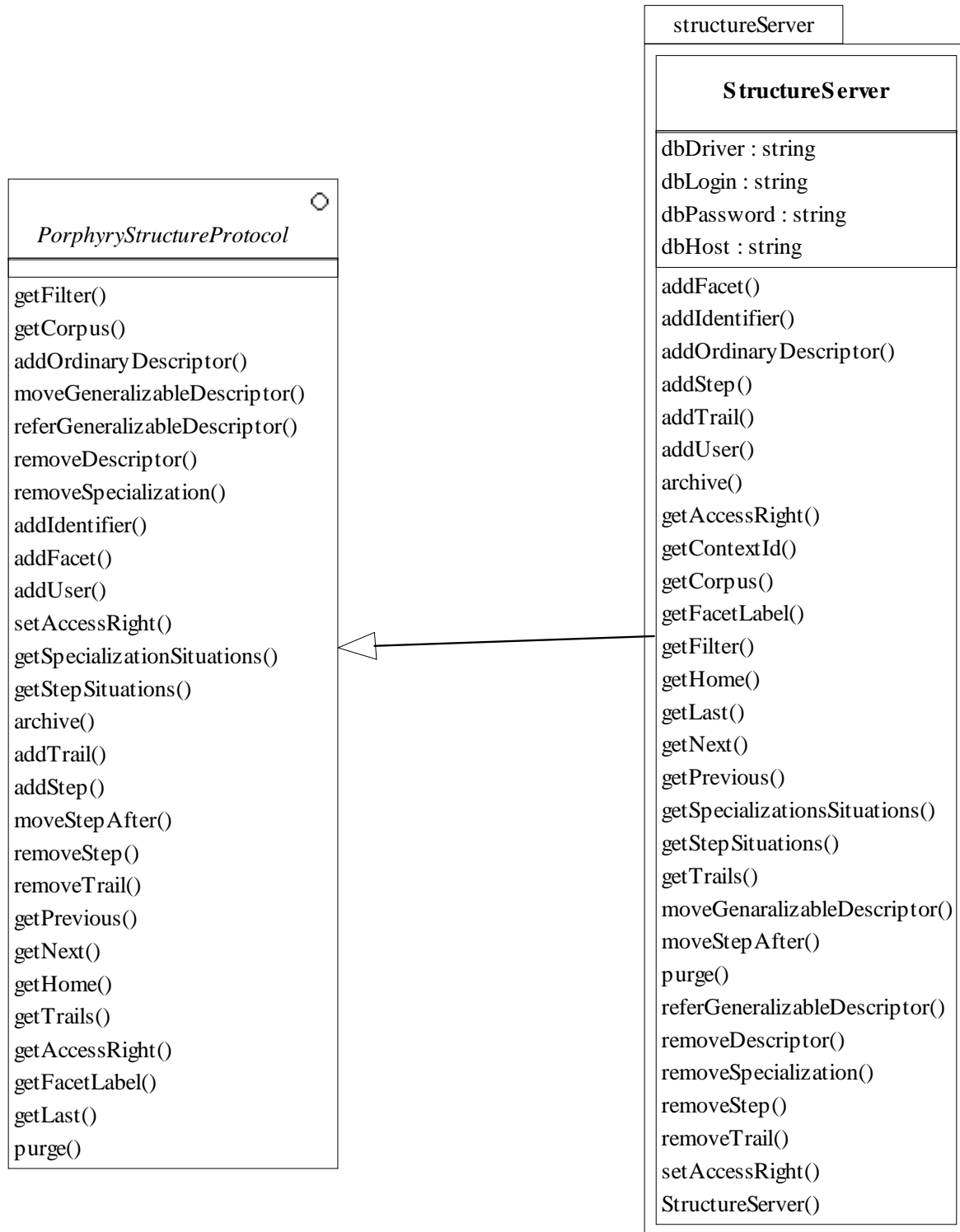
Comme cela a été annoncé dans l'introduction, ce document se situe aussi bien en aval qu'en amont de la réalisation proprement dite du serveur de structure. Cette partie concernant l'implémentation de la base de données a été, bien évidemment ajoutée après coup. En effet, une fois les fonctionnalités et le protocole mis en place par le serveur définis, il restait à étudier les différentes implémentations possibles de la base de données. En particulier, il fallait s'attacher à étudier la possibilité de mettre en place une architecture conservant les liens d'héritage entre les tables et la comparer ; en terme d'efficacité, de cohérence et de sûreté ; à une implémentation relationnelle plus classique.

Cette dernière solution a été choisie une fois sa faisabilité étudiée grâce au système de gestion de bases de données *postgresql*. Cette architecture s'est révélée solide et a permis, comme prévu, de respecter, point par point l'ensemble des concepts de la conception.





Structure Server



2.3.1.4. PorphyryStructureProtocol

Ces fonctions constituent l'interface partagée du serveur de structure ; elles sont appelées par le client et exécutées sur le serveur. Ces fonctions permettent de gérer les corpus de documents, les réseaux de description ainsi que les parcours de lecture.

L'Interface *PorphyryStructureProtocol* remplace l'ancienne interface 2002 *GraphServer*.

1. getFilter ()

Fonction de filtrage descendant récursivement dans les descripteurs connus d'une facette donnée et s'arrêtant aux descripteurs possibles et impossibles.

getFilter (In login : string, In password : string, In facetId : integer,
In corpus : Set) : [3] Set

Paramètres d'entrée :

- *facetId* : identifiant de la facette sur laquelle on travaille.
- *corpus* : Ensemble de localisations d'objets documentaires (*org.porphyry.shared.DocumentObjectLocator*).

Retour : Tableau de 3 sets

- Premier Set : ensemble de localisations d'objets documentaires (*org.porphyry.shared.DocumentObjectLocator*).
- Deuxième Set : ensemble de descripteurs avec leur état (*org.porphyry.shared.DescriptorWithState*).
- Troisième Set : ensemble de spécialisations (*org.porphyry.shared.Specialization*).

Ancien :

getGraphView (In sessionId : integer, In selectedCorpus : Set, In key : integer,
Out identifiersResult : Set, Out descriptorsResult : Set,
Out specializationsResult : Set, Out descriptorsDetailsResult : Set,
Out communitiesDetailsResult : Set,
Out specializationsDetailsResult : Set) : Set

Dans la version 2002 (*getGraphView*), le premier résultat donné par *getGraphView* (*identifiersResult*) était, puisque l'on n'avait pas de serveur de correspondance, une liste d'identifiants et non une liste de localisations d'objets documentaires (cette notion d'objet documentaire n'existant pas ...)

2. getCorpus ()

Fonction permettant par récursivité d'obtenir l'ensemble des objets documentaires décrits directement ou indirectement par une sélection de descripteurs donnée (ensemble de clefs de descripteurs).

getCorpus(In login : string , In password : string , In facetId : integer,
In descriptorSelection : Set) : Set

Paramètres d'entrée :

- *facetId* : identifiant de la facette sur laquelle travaille.
- *descriptorSelection* : Ensemble d'entiers dans lequel chaque élément correspond à la clef (key) d'un descripteur.

Retour : ensemble de localisations d'objets documentaires (*org.porphiry.shared.DocumentObjectLocator*), c'est à dire l'intersection des objets documentaires décrits directement ou indirectement par la sélection entrée.

Ancien :

getIdentifiersForAll (In selection : Collection) : Set

Le fonctionnement des fonctions `getCorpus ()` et `getFilter ()` est le suivant : les appels à `getCorpus` sont faits par le client qui transmet l'intersection des différents résultats à `getFilter`. Les sorties de `getCorpus` se retrouvent donc en entrée de `getFilter`. Sur l'ancien système, c'était `getGraphView` qui appelait les différents `getIdentifiersForAll` (ce qui explique qu'il n'y ait pas de paramètres de session dans cette fonction, ils étaient résolus avant) et on retrouvait les paramètres d'entrée de `getIdentifiersForAll` en paramètre d'entrée de `getGraphView`.

3. addOrdinaryDescriptor ()

Crée un ordinarydescriptor (généralisable et spécialisable) sous un generaldescriptor donné.

addOrdinaryDescriptor (In login : string, In password : string,
In generalDescriptor : integer, In label : string) : integer

Paramètres d'entrée :

- *generalDescriptor* : descripteur "specializable" sous lequel on va créer le nouveau descripteur.
- *label* : nom du nouveau descripteur.

Retour : entier représentant la clef (key) du descripteur créé.

Ancien :

addFolder(In sessionId : integer, In general : integer, In label : string, In author : string,
In creationDate : Date, In publisher : string, In publishingDate : Date,
In obsolator : string, In obsoletionDate : Date) : integer

4. moveGeneralizableDescriptor ()

Déplace toutes les spécialisations vers le descripteur donné et crée une nouvelle spécialisation sous le generaldescriptor donné.

moveGeneralizableDescriptor (In login : string, In password:string,
In generalDescriptor : integer,
In specialDescriptor : integer)

Paramètres d'entrée :

- *generalDescriptor* : identifiant du descripteur "specializable" sous lequel on va déplacer le descripteur.
- *specialDescriptor* : identifiant du descripteur que l'on souhaite déplacer.

Ancien :

move (In sessionId : integer, In general : integer, In special : integer, In author : string,
In creationDate : Date, In publisher : string, In publishingDate : Date,
In obsolator : string, In obsoletionDate : Date)

5. referGeneralizableDescriptor ()

Crée un lien de spécialisation entre deux descripteurs existant ; un "specializable" et un "generalizable".

referGeneralizableDescriptor (In login : string, In password : string,
In generalDescriptor : integer,
In specialDescriptor : integer)

Paramètres d'entrée :

- *generalDescriptor* : identifiant du descripteur "specializable", sommet du lien de spécialisation.
- *specialDescriptor* : identifiant du descripteur "generalizable", base du lien de spécialisation.

Ancien :

refer (In sessionId : integer, In general : integer, In special : integer, In author : string,
In creationDate : Date, In publisher : string, In publishingDate : Date,
In obsolator : string, In obsoletionDate : Date)

6. removeDescriptor ()

Efface tous les liens de spécialisation partant d'un descripteur et répète l'opération à ses descendants s'ils ne sont pas descendants d'autres descripteurs.

removeDescriptor (In login : string, In password : string, In descriptor : integer)

Paramètres d'entrée :

- *descriptor* : identifiant du descripteur constituant le "nœud" à partir duquel on efface.

Ancien :

removeFolder (In sessionId : integer, In key : integer)

7. removeSpecialization ()

Supprime un lien de spécialisation entre deux descripteurs donnés.

removeSpecialization (In login : string, In password : string,
In generalDescriptor : integer, In specialDescriptor : integer)

Paramètres d'entrée :

- *generalDescriptor* : identifiant du descripteur source du lien de spécialisation à supprimer.
- *specialDescriptor* : identifiant du descripteur cible du lien de spécialisation à supprimer.

Ancien :

void **removeSpecialization** (In sessionId : integer, In general : integer, In special : integer)

8. addIdentfier ()

Crée un nouvel identifiant sous un general descriptor donné de telle manière que cet identifiant référence l'objet documentaire donné.

addIdentfier (In login : string, In password : string, In generalDescriptor : integer,
In label : string, In documentObjectIP : InetAddress,
In documentObjectID : string) : integer

Paramètres d'entrée :

- *generalDescriptor* : identifiant du descripteur sous lequel on crée l'identifiant.
- *label* : description, nom du nouvel identifiant.
- *documentObjectIP* : adresse IP du serveur de correspondance contenant le document object.
- *documentObjectID* : nom logique ou aléatoire du document object.

Retour : entier représentant la clef (key) de l'identifiant créé.

Ancien :

addIdentfier (In sessionId : integer, In general : integer, In contentFile : string,
In fragmentParameters : string, In contentServer : string,
In fragmentServer : string, In descriptionServer : string,
In originalCommunityId : string, In originalId : integer,
In identifierType : integer, In noteText : string, In author : string,
In creationDate : Date, In publisher : string, In publishingDate : Date,
In obsolator : string, In obsoletionDate : Date) : integer

9. addFacet ()

Crée une nouvelle facette qui sera accessible en écriture pour son créateur.

addFacet (In login : string, In password : string, In label : string) : integer

Paramètres d'entrée :

- *label* : description, nom de la nouvelle facette.

Retour : entier représentant la clef (key) de la facette créée.

10. addUser ()

Crée un nouvel utilisateur.

addUser (In login : string, In password : string, In newLogin : string,
In newPassword : string, In newName : string)

Paramètres d'entrée :

- *newLogin* : login du nouvel utilisateur.
- *newPassword* : password du nouvel utilisateur.
- *newName* : nom du nouvel utilisateur.

11. setAccessRight ()

Enregistre les droits d'accès (lecture\écriture, lecture seule ou aucun) sur une facette donnée pour un utilisateur donné.

setAccessRight (In login : string, In password : string, In facetId : integer,
In userLogin : string, In canWrite : boolean, In canRead : boolean)

Paramètres d'entrée :

- *facetId* : identifiant de la facette concernée.
- *userLogin* : login de l'utilisateur concerné.
- *canWrite* : si vrai, l'utilisateur aura les droits d'écriture sur la facette.
- *canRead* : si vrai, l'utilisateur aura les droits de lecture sur la facette.

12. **getSpecializationSituations ()**

Donne l'historique d'une spécialisation donnée.

GetSpecializationSituations (In login : string, In password : string,
In generalDescriptor : integer,
In specialDescriptor : integer) : Set

Paramètres d'entrée :

- *generalDescriptor* : identifiant du descripteur source de la spécialisation.
- *specialDescriptor* : identifiant du descripteur cible de la spécialisation.

Retour : Ensemble de situations (*org.porphiry.shared.Situation*).

Ancien :

Cette fonction remplace la huitième partie du retour de la fonction `getGraphView`.

13. **getStepSituations ()**

Donne l'historique d'une étape donnée d'un parcours de lecture.

GetStepSituations (In login : string, In password : string, In trail : integer,
In documentObjectLocator : DocumentObjectLocator) : Set

Paramètres d'entrée :

- *trail* : identifiant du parcours de lecture.
- *documentObjectLocator* : localisation du document object constituant l'étape de lecture.

Retour : Ensemble de situations (*org.porphiry.shared.Situation*).

14. archive ()

Met à jour les contextes (spécialisations, étapes, etc.) de la facette donnée. L'espace du lecteur devient ainsi identique à l'espace du scripteur.

archive (In login : string, In password : string, In facetId : integer)

Paramètres d'entrée :

- *facetId* : identifiant de la facette concernée.

Ancien :

archive (In sessionId : integer)

15. addTrail ()

Crée un nouveau parcours de lecture avec le label donné sur une facette donnée.

addTrail (In login : string, In password : string, In facetId : integer,
In label : string) : integer

Paramètres d'entrée :

- *label* : description, nom du parcours de lecture.
- *facetId* : identifiant de la facette concernée.

Retour : identifiant numérique entier représentant la clef du parcours créé.

Ancien :

saveOnDBTrail (In trailLabel : string, In trail : Trail, In sessionId : integer)

16. addStep ()

Crée une nouvelle étape de lecture et la place à la fin du parcours de lecture donné.

addStep (In login : string, In password : string, In trailId : integer,
In documentObjectIP : InetAddress, In documentObjectID : string,
In corpus : Set)

Paramètres d'entrée :

- *trailId* : identifiant du parcours de lecture où ajouter l'étape.
- *documentObjectIP* : adresse IP du serveur de correspondance contenant le document object.
- *documentObjectID* : nom logique ou aléatoire du document object.
- *corpus* : Ensemble de localisations d'objets documentaires (*org.porphiry.shared.DocumentObjectLocator*).

Remarque : la clef de l'étape créée est (trailId, documentObjectIP, documentObjectID)

Ancien :

addStep (In trailId : integer, In stepOrder : integer, In step : Step, In sessionId : integer)

17. **moveStepAfter ()**

Déplace l'étape de lecture donnée (par son document object) derrière une autre étape du même parcours de lecture.

moveStepAfter (In login : string, In password : string, In trailId : integer,
In documentObjectIP : InetAddress, In documentObjectID : string,
In afterDocumentObjectIP : InetAddress,
In afterDocumentObjectID : string)

Paramètres d'entrée :

- *trailId* : identifiant du parcours de lecture concerné.
- *documentObjectIP* : adresse IP du serveur de correspondance contenant le document object de l'étape à déplacer.
- *documentObjectID* : nom logique ou aléatoire du document object de l'étape à déplacer.
- *afterdocumentObjectIP* : adresse IP du serveur de correspondance contenant le document object de l'étape derrière laquelle on désire se placer.
- *afterdocumentObjectID* : nom logique ou aléatoire du document object de l'étape derrière laquelle on désire se placer.

Remarque : Pour faire passer une étape en tête du parcours de lecture, les paramètres *afterDocumentObjectIP* et *afterDocumentObjectID* doivent être mis à *null*.

18. removeStep ()

Supprime une étape donnée (par son document object) d'un parcours de lecture donné.

removeStep (In login : string, In password : string, In trailId : integer,
In documentObjectIP : InetAddress, In documentObjectID : string)

Paramètres d'entrée :

- *trailId* : identifiant du parcours de lecture où supprimer l'étape.
- *documentObjectIP* : adresse IP du serveur de correspondance contenant le document object.
- *documentObjectID* : nom logique ou aléatoire du document object.

Ancien :

delStep (In trailId : integer, In stepToRemove : integer, In identifierIds : array,
In sessionId : integer)

19. removeTrail ()

Supprime le parcours de lecture donné.

removeTrail (In login : string, In password : string, In trailId : integer)

Paramètres d'entrée :

- *trailId* : identifiant du parcours de lecture à supprimer.

Ancien :

eraseOnDBTrail (In trailId : integer, In sessionId : integer)

20. `getPrevious ()`

Pour une étape de lecture donnée (par son document object), retourne l'étape de lecture précédente.

getPrevious (In login : string, In password : string, In trailId : integer,
In documentObjectIP : InetAddress,
In documentObjectID : string) : Step

Paramètres d'entrée :

- *trailId* : identifiant du parcours de lecture concerné.
- *documentObjectIP* : adresse IP du serveur de correspondance contenant le document object de l'étape courante.
- *documentObjectID* : nom logique ou aléatoire du document object de l'étape courante.

Retour : étape de lecture précédente (*org.porphiry.shared.Step*).

Remarque : retourne *null* si l'élément est le premier de la liste.

21. `getNext ()`

Pour une étape de lecture donnée (par son document object), retourne l'étape de lecture suivante.

getNext (In login : string, In password : string, In trailId : integer,
In documentObjectIP : InetAddress,
In documentObjectID : string) : Step

Paramètres d'entrée :

- *trailId* : identifiant du parcours de lecture concerné.
- *documentObjectIP* : adresse IP du serveur de correspondance contenant le document object de l'étape courante.
- *documentObjectID* : nom logique ou aléatoire du document object de l'étape courante.

Retour : localisation de l'étape de lecture suivante (*org.porphiry.shared.Step*).

Remarque : retourne *null* si l'élément est le dernier de la liste.

22. **getHome ()**

Pour un parcours de lecture donné, retourne la première étape de lecture.

getHome (In login : string, In password : string,
In trailId : integer) : Step

Paramètres d'entrée :

- *trailId* : identifiant du parcours de lecture concerné.

Retour : localisation de la première l'étape du parcours de lecture
(*org.porphiry.shared.Step*).

Ces opérations de navigation dans les parcours de lecture pouvaient s'effectuer, sur l'ancien système, en parcourant le vecteur décrivant un parcours de lecture obtenu de manière complète par les fonctions suivantes :

loadTrail (In myTrailId : integer, In sessionId : integer) : Trail

Récupérer un parcours de lecture.

getTrailIdList (In key : integer, In sessionId : integer) : Vector

Récupérer la liste des identifiants des parcours de lecture d'un utilisateur.

getTrailNameList (In key : integer, In sessionId : integer) : Vector

Récupérer le nom des parcours de lecture d'un utilisateur

getDescriptorLabel (In key : integer, In sessionId : integer) : string

Récupérer le label d'un descriptor.

23. getTrails ()

Pour une facette donnée, retourne tous les parcours comportant une étape contenant un documentObject donné.

getTrails (In login : string, In password : string,
In facetId : integer, In documentObjectIP : InetAddress,
In documentObjectID : string) : Set

Paramètres d'entrée :

- *facetId* : identifiant de la facette concernée.
- *documentObjectIP* : adresse IP du serveur de correspondance contenant le document object à rechercher.
- *documentObjectID* : nom logique ou aléatoire du document object à rechercher.

Retour : Ensemble des parcours de lectures satisfaisant la requête (*org.porphry.shared.Trail*).

24. getAccessRight ()

Pour une facette et un utilisateur donnés, retourne l'ensemble des droits d'accès de ce dernier sur la première.

getAccessRightTrails (In login:string ,In password:string ,In facetId:integer) :
AccessRight

Paramètres d'entrée :

- *facetId* : identifiant de la facette consultée.

Retour : Type de droits de l'utilisateur sur la facette (*org.porphry.shared.AccessRight*).

25. getFacetLabel ()

Cette fonction renvoie le label d'une facette donnée. Elle est utilisée pour afficher le nom de la facette avant que l'utilisateur ne s'y soit connectée. Ainsi, toute personne pour accéder, au moins, au nom de la facette.

getFacetLabel (In facetId:integer) : String

Paramètres d'entrée :

- *facetId* : identifiant de la facette consultée.

Retour : Description, label de la facette.

26. getLast ()

Pour un parcours de lecture donné, retourne la dernière étape de lecture

getLast (In login:string ,In password:string , In trailId:integer) : Step

Paramètres d'entrée :

- *trailId* : identifiant du parcours de lecture consulté.

Retour : dernière étape du parcours de lecture (*org.porphiry.shared.Step*).

27. purge ()

Pour une facette et un utilisateur donnés, supprime l'ensemble des objets qui ne sont plus ni privés ni publics.

getLast (In login:string ,In password:string , In facetId:integer)

Paramètres d'entrée :

- *facetId* : identifiant de la facette concernée.

2.3.2. Class StructureServer

La classe StructureServer est l'objet serveur accessible par le client grâce à la technologie RMI¹. Partant du principe qu'il peut exister plusieurs serveurs de structures physiquement distants contenant l'ensemble des structures créées par ses membres, il devient ainsi possible, pour un client, de se connecter à plusieurs structures éventuellement réparties sur différents serveurs. De la même façon, une structure peut être accédée par plusieurs clients.

Attributs :

L'ensemble des attributs de cette classe constitue les paramètres de connexion à la base de donnée du serveur hébergeant la classe. On y retrouve les identifiants et mots de passe de connexion (*dbLogin* et *dbPassword*) à la base de données ainsi que les références à l'hôte l'hébergeant et les drivers nécessaires (*dbHost* et *dbDriver*).

Méthodes :

Les deux méthodes spécifiées sont deux méthodes privées destinées essentiellement à factoriser un certain nombre de traitements effectués à chaque requête du client sur la base de données.

- **getAccessRight ()** : cette fonction sera chargée de récupérer les informations personnelles du client et d'en sortir les informations concernant les droits dont il dispose sur la facette considérée (aucun, lecture, lecture/écriture). En effet, puisque l'on souhaite désormais travailler en mode non connecté, cette vérification devra être réalisée à chaque requête. Cette fonction permet ainsi d'éviter la duplication du code.
- **executeQuery ()** : Encore une fois, il s'agit là de factoriser au maximum le code puisque cette fonction effectuera l'ensemble des traitements communs à toute requête SQL ; quelle que soit la méthode du serveur de structure effectuant cette requête.

¹ **Remote Method Invocation** : API Java permettant de manipuler des objets distants instanciés sur une autre machine de manière transparente pour l'utilisateur.

3. Synthèse

Ce document a donc accompagné la mise en place du serveur de structure depuis la première étude jusqu'à son fonctionnement pratique. L'état actuel reflète donc, à peu de chose près, et hors le travail de développement proprement dit, l'ensemble du travail et des réflexions menées autour de ce serveur.

La nouvelle architecture mise en place, aussi bien au niveau de la base de données qu'au niveau du serveur lui-même semblent fonctionner très correctement. Il paraissait effectivement nécessaire d'effectuer cette refonte d'architecture, surtout au niveau de l'organisation des classes. Le fonctionnement incrémental des années précédentes avait en effet naturellement entraîné une complexité énorme. Ce qui existe aujourd'hui semble être à la fois plus évolué, prenant appui, bien évidemment sur les travaux précédemment effectués, et plus simple au niveau de la compréhension du code et du fonctionnement général.

Annexe1 : Objets partagés

Ils peuvent être utilisés aussi bien par les clients que par les serveurs. Il s'agit de primitives, d'exceptions, de types, etc.

Org.porphry.shared.DocumentObjectLocator

Description complète d'un Document Object incluant sa localisation physique.

- **ip** : InetAddress : adresse IP du serveur de correspondance contenant le DocumentObject.
- **id** : string : nom logique ou aléatoire du DocumentObject (Ex : "BCH_121_3_493").

Org.porphry.shared.DescriptorWithState

Descripteur auquel on a associé un état pour un corpus donné.

- **key** : integer : clef numérique, identifiant du descripteur.
- **label** : string : description, nom usuel du descripteur.
- **state** : State : état du descripteur (voir plus loin).

State : type personnel entier modélisant l'état d'un descripteur pour un corpus donné. Il peut prendre les valeurs définies dans (org.porphry.shared.State).

Org.porphry.shared.Specialization

Décrit un descripteur en tant que spécialisation d'un descripteur "*specializable*".

- **general** : integer: identifiant du père du descripteur.
- **special** : integer : identifiant du descripteur.

Org.porphiry.shared.Situation

Décrit une situation dans son ensemble.

- **who** : string : acteur de la situation (Nom+Prénom).
- **when** : Date : date d'apparition de la situation.
- **what** : Action : opération effectuée (voir plus loin).

Action : type personnel modélisant la nature d'une opération. Elle peut prendre les valeurs décrites par le type (*org.porphiry.shared.Action*).

Org.porphiry.shared.Trail

Décrit un parcours de lecture.

- **key** : integer: identifiant du parcours de lecture.
- **label** : string : description, nom du parcours de lecture.

Remarque : il existe, bien entendu, d'autres objets partagés mais ils n'interviennent pas, du moins directement, dans le fonctionnement du serveur de structure et il n'apparaît donc pas utile de les intégrer à cette annexe.

Org.porphiry.shared.Step

Définit une étape de d'un parcours de lecture. Ces objets définissent de proche en proche un parcours de lecture. Un Step est un contexte de lecture ce qui permet d'en dresser un historique complet.

- **currentDOL** : DocumentObjectLocator : décrit l'objet documentaire associé.
- **corpus** : ensemble de DocumentObjectLocator : corpus documentaire ; ensemble d'objets documentaires ayant une relation avec le Step.

Org.porphry.shared.AccessRight

Cet objet met en place le type *droits d'accès* comme une énumération de couple de valeurs : un entier et un string de description. Un tel objet est utilisé chaque fois que le serveur doit contrôler les droits d'un utilisateur sur une facette.

Valeurs autorisées :

- **NONE (0)**: L'utilisateur n'a aucun droit sur cette facette.
- **READONLY (1)** : L'utilisateur n'a que les droits de lecture sur cette facette.
- **READWRITE (2)**: L'utilisateur a les droits de lecture et d'écriture sur cette facette.

Org.porphry.shared.State

Cet objet met en place le type *State* comme une énumération de couple de valeurs : un entier et un string de description. Un type *State* est associé à un descripteur uniquement dans un corpus donné. Il est important de réaliser qu'un descripteur n'a pas de *State* proprement dit mais uniquement dans le cadre d'un corpus sélectionné.

Valeurs autorisées :

- **IMPOSSIBLE (0)** : Le descripteur est impossible dans ce corpus ; le corpus du descripteur a une intersection vide avec le corpus général.
- **KNOWN (1)** : Le descripteur est connu dans ce corpus ; le corpus du descripteur est contenu dans le corpus général..
- **POSSIBLE (2)**: Dans tous les autres cas.

Org.porphry.shared.Action

Cet objet met en place le type *Action* comme une énumération de couple de valeurs : un entier et un string de description. Un type *Action* représente le type d'opération effectuée sur un *Context* (spécialisation ou étape de lecture).

Valeurs autorisées :

- **CREATING (0)**: L'opération est une création de contexte.
- **PUBLISHING (1)** : L'opération est une publication de contexte.
- **DELETING (2)** : L'opération est une suppression de contexte.

Remarque : il existe, bien entendu, d'autres objets partagés mais ils n'interviennent pas, du moins directement, dans le fonctionnement du serveur de structure et il n'apparaît donc pas utile de les intégrer à cette annexe.