# D4.1a - VISHNU Tasks Management Service Package Design

**COLLABORATORS**

| | | | |
|---|---|---|---|
| | *TITLE* :<br><br>D4.1a - VISHNU Tasks Management Service Package Design | | |
| *ACTION* | *NAME* | *DATE* | *SIGNATURE* |
| WRITTEN BY | Benjamin Isnard, Daouda Traoré, Eugène Pamba Capo-Chichi, Kevin Coulomb, and Ibrahima Cissé | March 14, 2011 | |

**REVISION HISTORY**

| NUMBER | DATE | DESCRIPTION | NAME |
|---|---|---|---|
| 1 | 18/03/2011 | Deliverable version | SysFera |

# Contents

# List of Figures

# Chapter 1

# Document presentation

## 1.1 Document objectives

This document presents the detailed internal design of the Users Management System (TMS) package. The purpose of this package is to handle all aspects of user management and session management within the VISHNU system. The functional and non-functional requirements for this package are those described in the referenced specification documents. The current document is part of the design phase of the software and therefore its main goal is to define the main components of the system architecture and their relationships.

## 1.2 Document structure

- Chapter 1 contains a brief overview of the document content.
- Chapter 2 contains a high-level overview of the system architecture.
- Chapter 3 describes the internal API used for remote procedure calls through SysFera-DS.
- Chapter 4 describes the internal class and data structures

## 1.3 References

- [D1.1a]: VISHNU General specifications
- [D1.1b]: VISHNU Spécifications techniques des besoins
- [D1.1c]: VISHNU API Detailed specifications

## 1.4 Acronyms

- **API**: Application programming interface
- **CLI**: Command line interface
- **DB**: DataBase
- **n/a**: Not Appliable (used for serializable capability in function descriptions)
- **SeD**: A Server Daemon is a SysFera-DS agent that provides services through the SysFera-DS API.
- **TMS**: Users management system
- **WS**: Web services

## 1.5 Glossary

- **Components**: the software components represents a library or an executable program that provides a given interface to other components or to end-users.
- **Serialized type**: this is a class of data (C++ Class) which instances can be serialized in a XML string before being sent over an API (to or from the API). The data is deserialized on the other side of the channel in order to re-build the same instance of the class.
- **SysFera-DS**: open-source middleware developped by SysFera.

# Chapter 2

# System Architecture

## 2.1 Overview of the TMS software infrastructure

We present in this section a detailed description of the TMS package architecture in terms of software components. In addition we show the dependencies between components to highlight their reuse. These components follow a client/server model. We present the different software layers from services (provided directly to the user) to the database (used by the server). The TMS client server package has been split into eight different interrelated components. The diagrams shown in section 2.3 describe the relationships between these components. The definitions of the components are the following:

- **External API** contains precisely the services provided to the user as defined in the detailed specifications. We're on the client side.
- **Internal API** is the middle layer of the server side. The services announced previously are performed here by combining a set of classes defined in the two following components.
- **TMS Client** contains intermediate (proxy) classes providing remote access to the business objects of **TMS Server**.
- **TMS Server** contains all classes implementing business objects by encapsulating the processing provided through the internal API.
- **Sysfera-DS Client API** is the C++ client API provided by the SysFera-DS toolbox.
- **Sysfera-DS Server API** is the C++ server API provided by the SysFera-DS toolbox.
- **Torque API** : the C API provided by the Torque batch scheduler.
- **LoadLeveler API** : the C API provided by the LoadLeveler batch scheduler..
- **Vishnu Database** stores all data manipulated by the TMS Server.

## 2.2 Deployment aspects of TMS

We explains here how the TMS package will be deployed in a physical hardware as illustrated in figure 2.1 where each cube represents an environement in which a component or a set of components execute. The TMS consists of:

- **TMS Server** is the provider of all TMS services. It consists of the TMS Monitor component and what we called the TMS SeD (TMS Server daemon) which gathers all TMS services published.
- **Client host** is TMS service requester. It contains all components allowing to make a TMS service request.
- **SysFera-DS Bus** is the specific software layer that ensures the communication between client hosts and server hosts.
- **Vishnu database**: this component represents a unique instance of an Oracle or PostgreSQL database.

It is important to note that we can have several Secondary TMS servers (where an TMS Sed is running in each) but only one instance of TMS Monitor daemon running in the Main TMS Server as we can see in figure 2.1.

## 2.3 Architecture diagrams

# Chapter 3

# Internal API specification

## 3.1 Generic definition formats presentation

This section presents the formats used in this chapter to describe the services provided by the internal API.

### 3.1.1 Service definition format

**Access**

Here is detailed the access level of the service 'myService' (i.e. the privilege required to use it)

**Parameters**

The following table contains all the input and output parameters of the service, along with their type and description.

| Parameter | Type | Serialized type | Description | Mode |
|---|---|---|---|---|
| sessionKey | string | n/a | This is an example of a required string input parameter | IN |
| listOfJobs | string | ListJobs | This is an example of an object output parameter that is serialized as a string | OUT |

**Description**

Here is detailed the purpose of the service 'myService'

**Return Value**

Here are detailed the different return codes provided by the service.

| Name | Description |
|---|---|
| VISHNU_OK | The service has been performed successfully. |
| TMS_UNKNOWN_MACHINE | This is the human-readable generic message that will be available to the user of the API. |

**Used by this(these) API function(s):**

This shows the list of functions from the external Vishnu API (see [D1_1c]) that use this service.

## 3.2 Definition of the services of the package

### 3.2.1 Service jobSubmit

**Access**

This service can be used by any VISHNU user

**Parameters**

| Parameter | Type | Serialized type | Description | Mode |
|---|---|---|---|---|
| sessionKey | string | n/a | The session key is the encrypted identifier of the session generated by VISHNU | IN |
| machineId | string | n/a | Is the id of the machine where the job must be submitted | IN |
| scriptFilePath | string | n/a | The path to the file containing the characteristics (job command, and batch scheduler directive required or optional) of the job to submit. | IN |
| options | string | SubmitOptions | Is an instance of the class SubmitOptions. It allows the user to submit job by using different options | IN |
| jobId | string | n/a | Is the returned id of the submitted job | OUT |
| jobPath | string | n/a | Is the path to the file containing job characteristics | OUT |
| errorInfo | string | n/a | Additional information provided when an error code is returned | OUT |

**Description**

The jobSubmit() function submits job on a machine through the use of a script (scriptFilePath).

**Return Value**

An error code is returned when an error occurs during the execution of the service

| Name | Description |
|---|---|
| VISHNU_OK | The service was performed successfully. |
| TMS_UNKNOWN_MACHINE | The machine is not known. |
| TMS_BATCH_SCHEDULER_ERROR | Indicates an error caused by the underlying batch scheduler |
| TMS_INVALID_PATH | The path to the file containing the characteristics of the job to submit is not a valid path |
| TMS_INVALID_RESPONSE | Indicates that the implementation produced a response that does not match the criteria defined by the specification. |
| TMS_INVALID_REQUEST | Indicates that the request is not valid. |
| TMS_INVALID_SESSION_KEY | The session key is not valid to perform the service. |
| TMS_PERMISSION_DENIED | Indicates the requested operation is not allowed for provided user. |
| TMS_SERVER_NOT_AVAILABLE | Indicates that the task management server is not available. |
| TMS_SUBMIT_SERVICE_NOT_AVAILABLE | Indicates that the service to perform the submit operation is not found. |
| TMS_UNKNOWN_BATCH_SCHEDULER_TYPE | Indicates that the batch scheduler type is not known. |
| TMS_UNKNOWN_QUEUE | Indicates that the specified queue by the user is not known. |
| DB_ERROR | A problem occurs with the database |

**Used by this(these) API function(s):**

None

### 3.2.2 Service jobCancel

**Access**

This service can be used by any VISHNU user

**Parameters**

| Parameter | Type | Serialized type | Description | Mode |
|---|---|---|---|---|
| sessionKey | string | n/a | The session key | IN |
| machineId | string | n/a | Machine hash key | IN |
| jobId | string | n/a | The Id of the job | IN |
| infoMsg | string | n/a | The information message | OUT |
| errorInfo | string | n/a | Additional information provided when an error code is returned | OUT |

**Description**

The jobCancel() function cancels a job from its id

**Return Value**

An error code is returned when an error occurs during the execution of the service

| Name | Description |
|---|---|
| TMS_BATCH_SCHEDULER_ERROR | Indicates an error caused by the underlying batch scheduler |
| TMS_INVALID_RESPONSE | Indicates that the implementation produced a response that does not match the criteria defined by the specification. |
| TMS_INVALID_REQUEST | Indicates that the request is not valid. |
| TMS_INVALID_SESSION_KEY | The session key is not valid to perform the service. |
| TMS_PERMISSION_DENIED | Indicates the requested operation is not allowed for provided user. |
| VISHNU_OK | The service was performed successfully. |
| TMS_SERVER_NOT_AVAILABLE | Indicates that the task management server is not available. |
| TMS_UNKNOWN_BATCH_SCHEDULER_TYPE | Indicates that the batch scheduler type is not known. |
| TMS_UNKNOWN_MACHINE | The machine is not known. |
| DB_ERROR | A problem occurs with the database |

**Used by this(these) API function(s):**

None

### 3.2.3 Service jobQueryGetInfoOfJob

**Access**

This service can be used by any VISHNU user

**Parameters**

| Parameter | Type | Serialized type | Description | Mode |
|---|---|---|---|---|
| sessionKey | string | n/a | The session key | IN |
| machineId | string | n/a | Machine hash key | IN |
| jobId | string | n/a | The id of the job | IN |
| jobInfos | string | Job | The resulting information on the job | OUT |
| errorInfo | string | n/a | Additional information provided when an error code is returned | OUT |

**Description**

The jobQueryGetInfoOfJob() function gets information on a job from its id

**Return Value**

An error code is returned when an error occurs during the execution of the service

| Name | Description |
|---|---|
| TMS_BATCH_SCHEDULER_ERROR | Indicates an error caused by the underlying batch scheduler |
| TMS_INVALID_REQUEST | Indicates that the request is not valid. |
| TMS_INVALID_RESPONSE | Indicates that the implementation produced a response that does not match the criteria defined by the specification. |
| TMS_INVALID_SESSION_KEY | The session key is not valid to perform the service. |
| TMS_PERMISSION_DENIED | Indicates the requested operation is not allowed for provided user. |
| VISHNU_OK | The service was performed successfully. |
| TMS_SERVER_NOT_AVAILABLE | Indicates that the task management server is not available. |
| TMS_UNKNOWN_BATCH_SCHEDULER_TYPE | Indicates that the batch scheduler type is not known. |
| TMS_UNKNOWN_MACHINE | The machine is not known. |
| DB_ERROR | A problem occurs with the database |

**Used by this(these) API function(s):**

None

### 3.2.4 Service jobQueryGetListOfJobs

**Access**

This service can be used by any VISHNU user

**Parameters**

| Parameter | Type | Serialized type | Description | Mode |
|---|---|---|---|---|
| sessionKey | string | n/a | The session key | IN |
| machineId | string | n/a | Machine hash key | IN |
| options | string | ListJobsOptions | Additional options for jobs listing | IN |
| listOfJobs | string | ListJobs | The constructed object list of jobs | OUT |
| errorInfo | string | n/a | Additional information provided when an error code is returned | OUT |

**Description**

The jobQueryGetListOfJobs() function gets a list of all submitted jobs

**Return Value**

An error code is returned when an error occurs during the execution of the service

| Name | Description |
|---|---|
| TMS_BATCH_SCHEDULER_ERROR | Indicates an error caused by the underlying batch scheduler |
| TMS_INVALID_RESPONSE | Indicates that the implementation produced a response that does not match the criteria defined by the specification. |
| TMS_INVALID_REQUEST | Indicates that the request is not valid. |
| TMS_INVALID_SESSION_KEY | The session key is not valid to perform the service. |
| TMS_PERMISSION_DENIED | Indicates the requested operation is not allowed for provided user. |
| VISHNU_OK | The service was performed successfully. |

| Name | Description |
|---|---|
| TMS_SERVER_NOT_AVAILABLE | Indicates that the task management server is not available. |
| TMS_UNKNOWN_BATCH_SCHEDULER_TYPE | Indicates that the batch scheduler type is not known. |
| TMS_UNKNOWN_MACHINE | The machine is not known. |
| DB_ERROR | A problem occurs with the database |

**Used by this(these) API function(s):**

None

### 3.2.5 Service jobResultGetOutPutOfJob

**Access**

This service can be used by any VISHNU user

**Parameters**

| Parameter | Type | Serialized type | Description | Mode |
|---|---|---|---|---|
| sessionKey | string | n/a | The session key | IN |
| machineId | string | n/a | Machine hash key | IN |
| jobId | string | n/a | The Id of the job | IN |
| outputPath | string | n/a | The path of the file containinig the output result of the job | OUT |
| errorPath | string | n/a | The path of the file containinig the errors that has been occurred during the execution of the job | OUT |
| errorInfo | string | n/a | Additional information provided when an error code is returned | OUT |

**Description**

The jobResultGetOutPutOfJob() function gets outputPath and errorPath of a job from its id

**Return Value**

An error code is returned when an error occurs during the execution of the service

| Name | Description |
|---|---|
| TMS_BATCH_SCHEDULER_ERROR | Indicates an error caused by the underlying batch scheduler |
| TMS_INVALID_RESPONSE | Indicates that the implementation produced a response that does not match the criteria defined by the specification. |
| TMS_INVALID_REQUEST | Indicates that the request is not valid. |
| TMS_INVALID_SESSION_KEY | The session key is not valid to perform the service. |
| VISHNU_OK | The service was performed successfully. |
| TMS_PERMISSION_DENIED | Indicates the requested operation is not allowed for provided user. |
| TMS_SERVER_NOT_AVAILABLE | Indicates that the task management server is not available. |
| TMS_UNKNOWN_BATCH_SCHEDULER_TYPE | Indicates that the batch scheduler type is not known. |
| TMS_UNKNOWN_MACHINE | The machine is not known. |
| DB_ERROR | A problem occurs with the database |

**Used by this(these) API function(s):**

None

### 3.2.6 Service jobResultGetAllJobsOutPut

**Access**

This service can be used by any VISHNU user

**Parameters**

| Parameter | Type | Serialized type | Description | Mode |
|---|---|---|---|---|
| sessionKey | string | n/a | The session key | IN |
| machineId | string | n/a | Machine hash key | IN |
| listOfResults | string | ListJobResults | Is the list of jobs results | OUT |
| errorInfo | string | n/a | Additional information provided when an error code is returned | OUT |

**Description**

The jobResultGetAllJobsOutPut() function dynamically gets outputPath and errorPath of completed jobs

**Return Value**

An error code is returned when an error occurs during the execution of the service

| Name | Description |
|---|---|
| TMS_BATCH_SCHEDULER_ERROR | Indicates an error caused by the underlying batch scheduler |
| TMS_INVALID_RESPONSE | Indicates that the implementation produced a response that does not match the criteria defined by the specification. |
| TMS_INVALID_REQUEST | Indicates that the request is not valid. |
| TMS_INVALID_SESSION_KEY | The session key is not valid to perform the service. |
| VISHNU_OK | The service was performed successfully. |
| TMS_PERMISSION_DENIED | Indicates the requested operation is not allowed for provided user. |
| TMS_SERVER_NOT_AVAILABLE | Indicates that the task management server is not available. |
| TMS_UNKNOWN_BATCH_SCHEDULER_TYPE | Indicates that the batch scheduler type is not known. |
| TMS_UNKNOWN_MACHINE | The machine is not known. |
| DB_ERROR | A problem occurs with the database |

**Used by this(these) API function(s):**

None

### 3.2.7 Service jobResultRefreshPeriodSet

**Access**

This service can be used by ADMIN users only

**Parameters**

| Parameter | Type | Serialized type | Description | Mode |
|---|---|---|---|---|
| sessionKey | string | n/a | The session key | IN |
| machineId | string | n/a | The id of the machine | IN |
| value | ##TBD## | n/a | Is the refresh interval value (in seconds) | IN |
| errorInfo | string | n/a | Additional information provided when an error code is returned | OUT |

**Description**

The jobResultRefreshPeriodSet() function sets the refresh period of output and error files contents

**Return Value**

An error code is returned when an error occurs during the execution of the service

| Name | Description |
|---|---|
| VISHNU_OK | The service was performed successfully. |
| TMS_UNKNOWN_MACHINE | The machine is not known. |

**Used by this(these) API function(s):**

None

### 3.2.8 Service jobProgressionGetResults

**Access**

This service can be used by any VISHNU user

**Parameters**

| Parameter | Type | Serialized type | Description | Mode |
|---|---|---|---|---|
| sessionKey | string | n/a | The session key | IN |
| machineId | string | n/a | Is the id of the machine to get the jobs progression. | IN |
| options | string | ProgressOptions | Is an object containing the available options jobs for progression . | IN |
| progress | string | Progression | Is the object containing jobs progression information | OUT |
| errorInfo | string | n/a | Additional information provided when an error code is returned | OUT |

**Description**

The jobProgressionGetResults() function gets the progression status of jobs

**Return Value**

An error code is returned when an error occurs during the execution of the service

| Name | Description |
|---|---|
| VISHNU_OK | The service was performed successfully. |
| TMS_INVALID_REQUEST | Indicates that the request is not valid. |
| TMS_UNKNOWN_MACHINE | The machine is not known. |
| TMS_INVALID_SESSION_KEY | The session key is not valid to perform the service. |
| TMS_SERVER_NOT_AVAILABLE | Indicates that the task management server is not available. |
| TMS_SUBMIT_SERVICE_NOT_AVAILABLE | Indicates that the service to perform the submit operation is not found. |
| DB_ERROR | A problem occurs with the database |

**Used by this(these) API function(s):**

None

### 3.2.9 Service QueueList

**Access**

This service can be used by any VISHNU user

**Parameters**

| Parameter | Type | Serialized type | Description | Mode |
|---|---|---|---|---|
| sessionKey | string | n/a | The session key | IN |
| machineId | string | n/a | Machine hash key | IN |
| listofQueues | string | ListQueues | The list of queues | OUT |
| errorInfo | string | n/a | Additional information provided when an error code is returned | OUT |

**Description**

The QueueList() function gets queues information

**Return Value**

An error code is returned when an error occurs during the execution of the service

| Name | Description |
|---|---|
| TMS_BATCH_SCHEDULER_ERROR | Indicates an error caused by the underlying batch scheduler |
| TMS_INVALID_RESPONSE | Indicates that the implementation produced a response that does not match the criteria defined by the specification. |
| TMS_INVALID_REQUEST | Indicates that the request is not valid. |
| TMS_INVALID_SESSION_KEY | The session key is not valid to perform the service. |
| TMS_PERMISSION_DENIED | Indicates the requested operation is not allowed for provided user. |
| VISHNU_OK | The service was performed successfully. |
| TMS_SERVER_NOT_AVAILABLE | Indicates that the task management server is not available. |
| TMS_UNKNOWN_BATCH_SCHEDULER_TYPE | Indicates that the batch scheduler type is not known. |
| TMS_UNKNOWN_MACHINE | The machine is not known. |
| DB_ERROR | A problem occurs with the database |

**Used by this(these) API function(s):**

None

# Chapter 4

# Internal class and data structures

## 4.1 Introduction

This chapter introduces the details of the implementation of the different components described in chapter 2 (Architecture). It is composed of three sections:

- **Client modelization**: describes the classes used to implement the *TMS Client* component.
- **Server modelization**: describes the classes used to implement the *TMS Server* component.
- **Data modelization**: describes the data structure used to implement the *TMS Client* component and the *TMS Server* component.
- **Vishnu core functions modelization**: describes the classes and data structures used to implement the the VISHNU cross-modules components (components that are used by TMS and other VISHNU modules).

## 4.2 TMS client modelization

### 4.2.1 Class diagrams

#### 4.2.1.1 UMS Client Class Diagram

This diagram describes all classes used to communicate with VISHNU System. Each class proxy contains the corresponding data class illustrated on the UMS Data modelization section and the methods usable by the UMS Client component. A QueryProxy class implements a generic model to list objects of VISHNU.

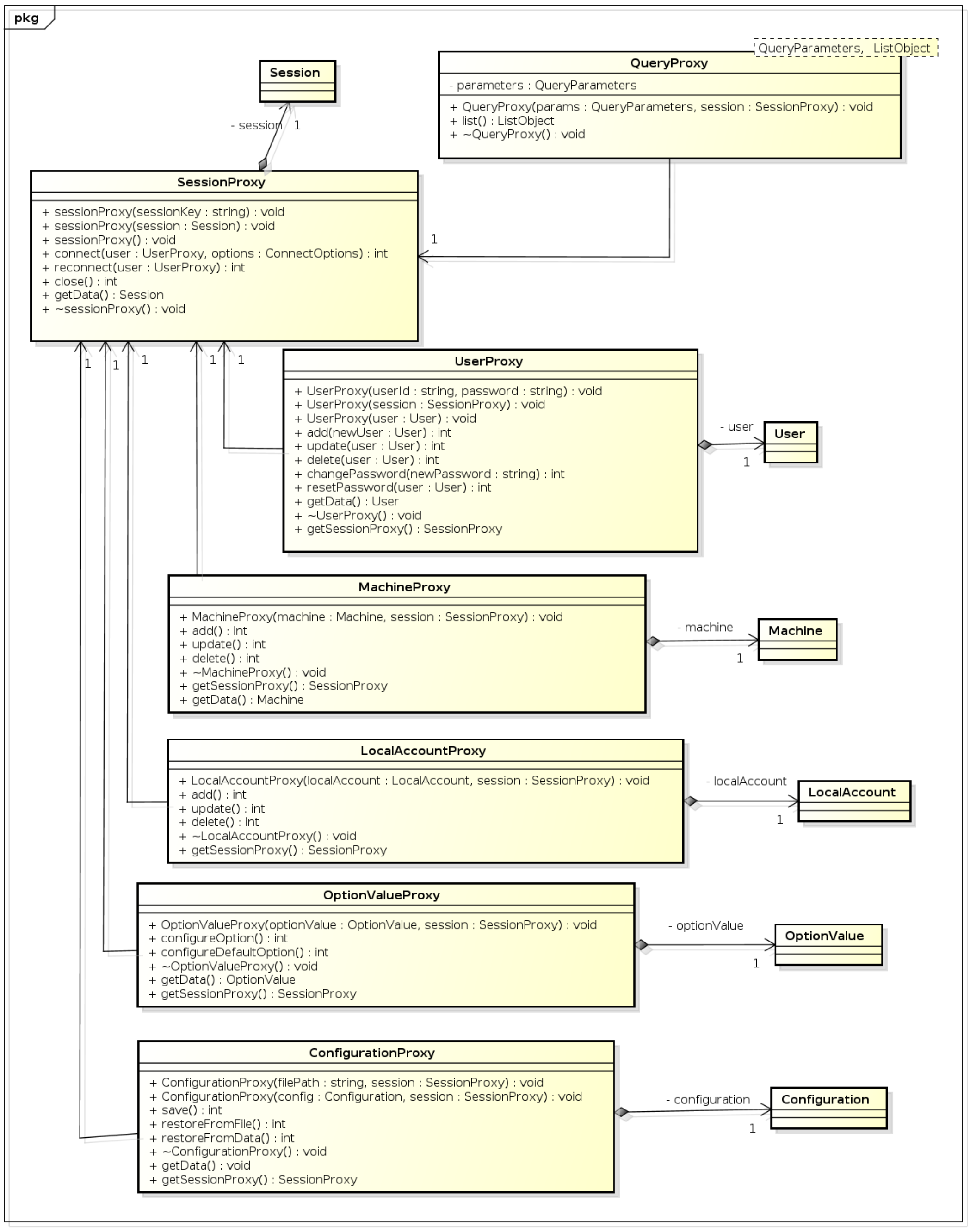
pkg
Session
- session
1
QueryProxy
QueryParameters, ListObject
- parameters : QueryParameters
+ QueryProxy(params : QueryParameters, session : SessionProxy) : void
+ list() : ListObject
+ ~QueryProxy() : void
1
SessionProxy
+ sessionProxy(sessionKey : string) : void
+ sessionProxy(session : Session) : void
+ sessionProxy() : void
+ connect(user : UserProxy, options : ConnectOptions) : int
+ reconnect(user : UserProxy) : int
+ close() : int
+ getData() : Session
+ ~sessionProxy() : void
1
1
1
1
1
UserProxy
+ UserProxy(userId : string, password : string) : void
+ UserProxy(session : SessionProxy) : void
+ UserProxy(user : User) : void
+ add(newUser : User) : int
+ update(user : User) : int
+ delete(user : User) : int
+ changePassword(newPassword : string) : int
+ resetPassword(user : User) : int
+ getData() : User
+ ~UserProxy() : void
+ getSessionProxy() : SessionProxy
- user
User
1
MachineProxy
+ MachineProxy(machine : Machine, session : SessionProxy) : void
+ add() : int
+ update() : int
+ delete() : int
+ ~MachineProxy() : void
+ getSessionProxy() : SessionProxy
+ getData() : Machine
- machine
Machine
1
LocalAccountProxy
+ LocalAccountProxy(localAccount : LocalAccount, session : SessionProxy) : void
+ add() : int
+ update() : int
+ delete() : int
+ ~LocalAccountProxy() : void
+ getSessionProxy() : SessionProxy
- localAccount
LocalAccount
1
OptionValueProxy
+ OptionValueProxy(optionValue : OptionValue, session : SessionProxy) : void
+ configureOption() : int
+ configureDefaultOption() : int
+ ~OptionValueProxy() : void
+ getData() : OptionValue
+ getSessionProxy() : SessionProxy
- optionValue
OptionValue
1
ConfigurationProxy
+ ConfigurationProxy(filePath : string, session : SessionProxy) : void
+ ConfigurationProxy(config : Configuration, session : SessionProxy) : void
+ save() : int
+ restoreFromFile() : int
+ restoreFromData() : int
+ ~ConfigurationProxy() : void
+ getData() : void
+ getSessionProxy() : SessionProxy
- configuration
Configuration
1

Figure 4.1: UMS Client Class Diagram

## 4.3 TMS server modelization

### 4.3.1 Class diagrams

#### 4.3.1.1 UMS Server Class Diagram

This diagram presents the main objects used by UMS server component to process the UMS Client component requests. Each object that can be listed have a static method list with the corresponding options.

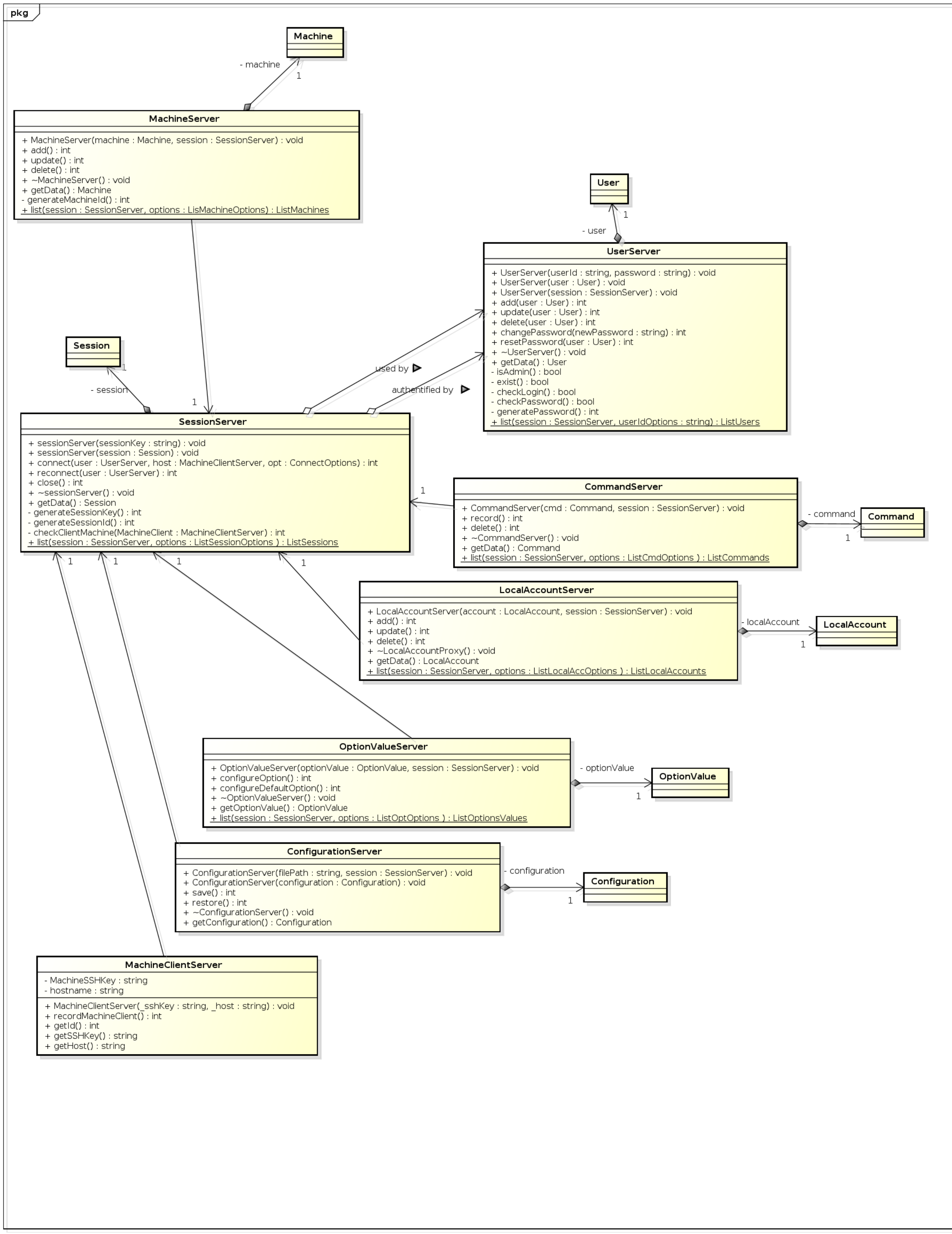
pkg
Machine
- machine
1
MachineServer
+ MachineServer(machine : Machine, session : SessionServer) : void
+ add() : int
+ update() : int
+ delete() : int
+ ~MachineServer() : void
+ getData() : Machine
- generateMachineId() : int
+ list(session : SessionServer, options : LisMachineOptions) : ListMachines
User
1
- user
UserServer
+ UserServer(userId : string, password : string) : void
+ UserServer(user : User) : void
+ UserServer(session : SessionServer) : void
+ add(user : User) : int
+ update(user : User) : int
+ delete(user : User) : int
+ changePassword(newPassword : string) : int
+ resetPassword(user : User) : int
+ ~UserServer() : void
+ getData() : User
- isAdmin() : bool
- exist() : bool
- checkLogin() : bool
- checkPassword() : bool
- generatePassword() : int
+ list(session : SessionServer, userIdOptions : string) : ListUsers
Session
1
- session
1
used by
authentified by
SessionServer
+ sessionServer(sessionKey : string) : void
+ sessionServer(session : Session) : void
+ connect(user : UserServer, host : MachineClientServer, opt : ConnectOptions) : int
+ reconnect(user : UserServer) : int
+ close() : int
+ ~sessionServer() : void
+ getData() : Session
- generateSessionKey() : int
- generateSessionId() : int
- checkClientMachine(MachineClient : MachineClientServer) : int
+ list(session : SessionServer, options : ListSessionOptions ) : ListSessions
1
CommandServer
+ CommandServer(cmd : Command, session : SessionServer) : void
+ record() : int
+ delete() : int
+ ~CommandServer() : void
+ getData() : Command
+ list(session : SessionServer, options : ListCmdOptions ) : ListCommands
- command
Command
1
1
1
1
1
LocalAccountServer
+ LocalAccountServer(account : LocalAccount, session : SessionServer) : void
+ add() : int
+ update() : int
+ delete() : int
+ ~LocalAccountProxy() : void
+ getData() : LocalAccount
+ list(session : SessionServer, options : ListLocalAccOptions ) : ListLocalAccounts
- localAccount
LocalAccount
1
OptionValueServer
+ OptionValueServer(optionValue : OptionValue, session : SessionServer) : void
+ configureOption() : int
+ configureDefaultOption() : int
+ ~OptionValueServer() : void
+ getOptionValue() : OptionValue
+ list(session : SessionServer, options : ListOptOptions ) : ListOptionsValues
- optionValue
OptionValue
1
ConfigurationServer
+ ConfigurationServer(filePath : string, session : SessionServer) : void
+ ConfigurationServer(configuration : Configuration) : void
+ save() : int
+ restore() : int
+ ~ConfigurationServer() : void
+ getConfiguration() : Configuration
- configuration
Configuration
1
MachineClientServer
- MachineSSHKey : string
- hostname : string
+ MachineClientServer(_sshKey : string, _host : string) : void
+ recordMachineClient() : int
+ getId() : int
+ getSSHKey() : string
+ getHost() : string

Figure 4.2: UMS Server Class Diagram

## 4.4 TMS data modelization

### 4.4.1 Class diagrams

#### 4.4.1.1 UMS Data Class Diagram

This diagram illustrates the structure and the relationship between data manipulated by the components Client and Server.

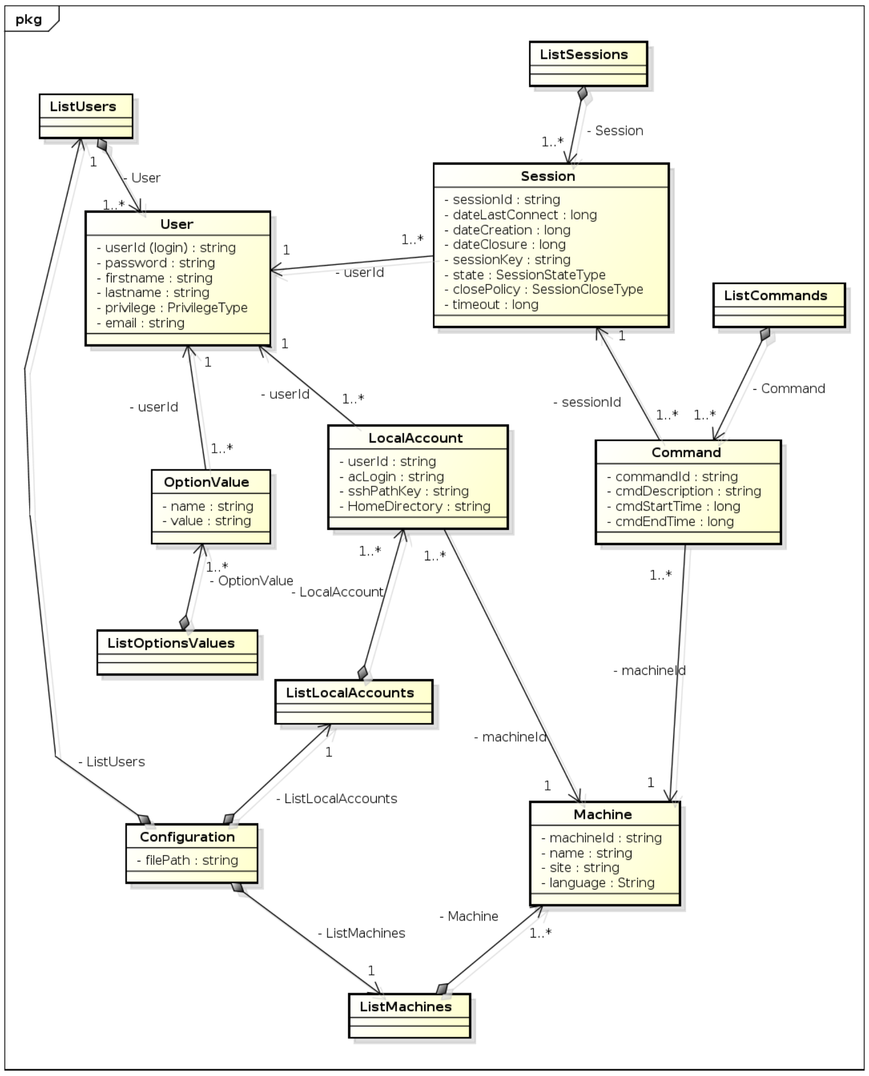

Figure 4.3: UMS Data Class Diagram

## 4.5 Vishnu core functions modelization

### 4.5.1 Introduction

The following elements describe the core classes (i.e. the classes that will be used by each module such as the exceptions and the databases). The modelization diagrams are given with some explanations about them.

### 4.5.2 Tables relationships

In order to have a coherent System, we have designed a relational model for the database. We need only one database that can contain all the Vishnu tables. The model is represented in figure 4.4. The rectangles are the tables and the lines represent the links between the tables.

The links between the tables are based on the following rules:

- - The *VISHNU* table has one or more *MACHINE*
  - A *MACHINE* is in one and only one *VISHNU* infrastructure
- - A *MACHINE* has one or more *CPU*
  - A *CPU* is in one and only one *MACHINE*
- - A *MACHINE* has one or more *DESCRIPTION*
  - A *DESCRIPTION* is for one and only one *MACHINE*
- - A *MACHINE* has one or more *THRESHOLD*
  - A *THRESHOLD* is for one and only one *MACHINE*
- - A *MACHINE* has one or more *ACCOUNT*
  - An *ACCOUNT* is for one and only one *MACHINE*
- - The *VISHNU* table has one or more *USER*
  - An *USER* is in one and only one *VISHNU* infrastructure
- - An *USER* has one or more *ACCOUNT*
  - An *ACCOUNT* is for one and only one *USER*
- - An *USER* has one or more *FILE TRANSFER*
  - A *FILE TRANSFER* is for one and only one *USER*
- - An *USER* has one or more *OPTION VALUE*
  - An *OPTION VALUE* is for one and only one *USER*
- - An *USER* sets one or more *THRESHOLD*
  - A *THRESHOLD* is set by one and only one *USER*
- - An *OPTION* has one or more *OPTION VALUE*
  - An *OPTION VALUE* is for one and only one *OPTION*
- - An *USER* has one or more *SESSION*
  - A *SESSION* is for one and only one *USER*
- - A *SESSION* has one or more *COMMAND*
  - A *COMMAND* is for one and only one *SESSION*
- - A *CLIENT MACHINE* has one or more *SESSION*
  - A *SESSION* is for one and only one *CLIENT MACHINE*
- - A *MACHINE* has one or more *STATE*
  - A *STATE* is for one and only one *MACHINE*
- - A *COMMAND* can have one or more *JOB*
  - A *JOB* is for one and only one *COMMAND*
- - A *COMMAND* can have one or more *FILE*
  - A *FILE* is for one and only one *COMMAND*

### 4.5.3 Relational model

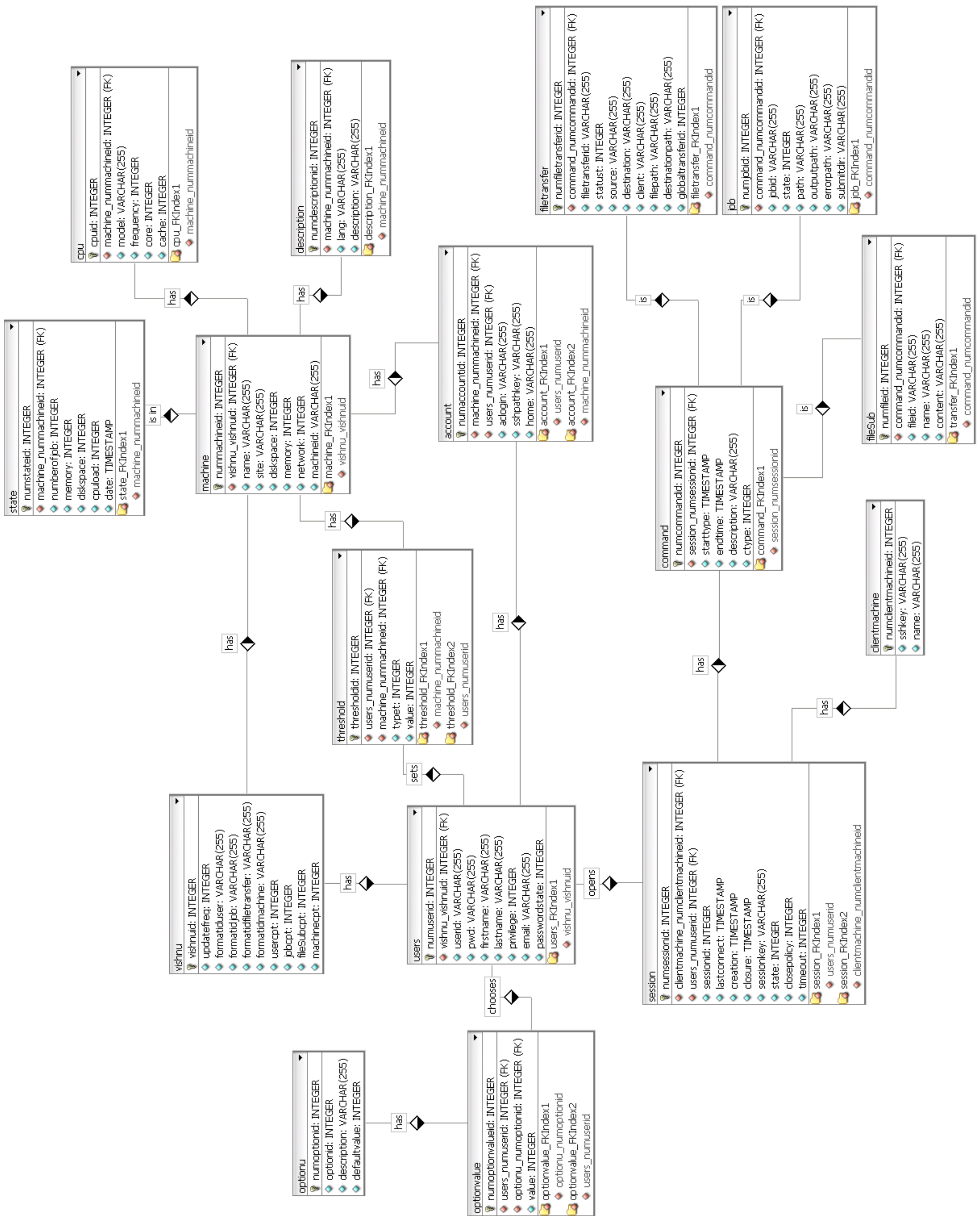

Figure 4.4: Relational model

### 4.5.4 The modelization

#### 4.5.4.1 The database classes

The database class diagram is very simple. There is a database interface that defines a set of public operations that can be done over a database:

- commit
- rollback
- execute a query
- connect
- disconnect ...

And there are two examples of classes that implement the database. There is also a factory that can create the databases. See the diagram 4.5.

#### 4.5.4.2 The exception classes

The exception class diagram defines a generic exception class, *VishnuException* that represents a generic exception that can be raised by a Vishnu function. This class has two subclasses, the *SystemException* that represents an exception due to a system problem and the *UserException* that represents an exception due to the user of the function (bad parameters typically). Both the server and clients have this way of building the exceptions. The *SystemException* has more specific subclasses depending on the modules that raises them. A key function is the append one, that allows to add a message to an existing vishnu exception. Thus, crossing the various level of the call can append information messages to specify the context of the exception. See the diagram 4.6.

### 4.5.5 Class diagrams

#### 4.5.5.1 DB class diagram

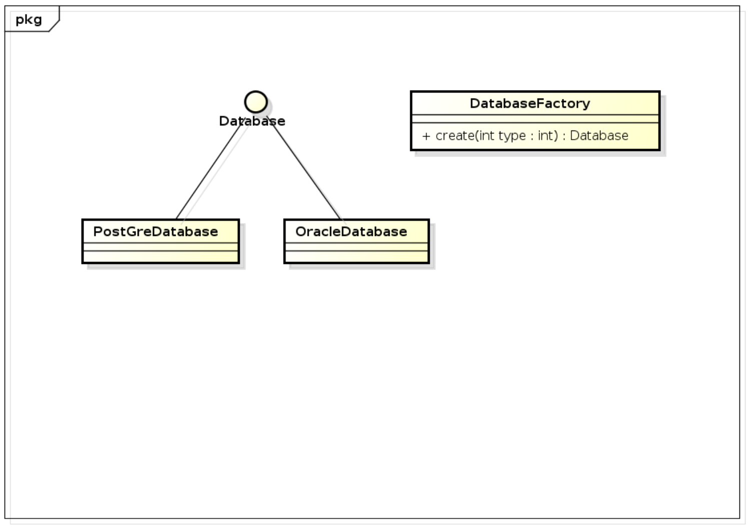

Figure 4.5: DB class diagram

#### 4.5.5.2 exception

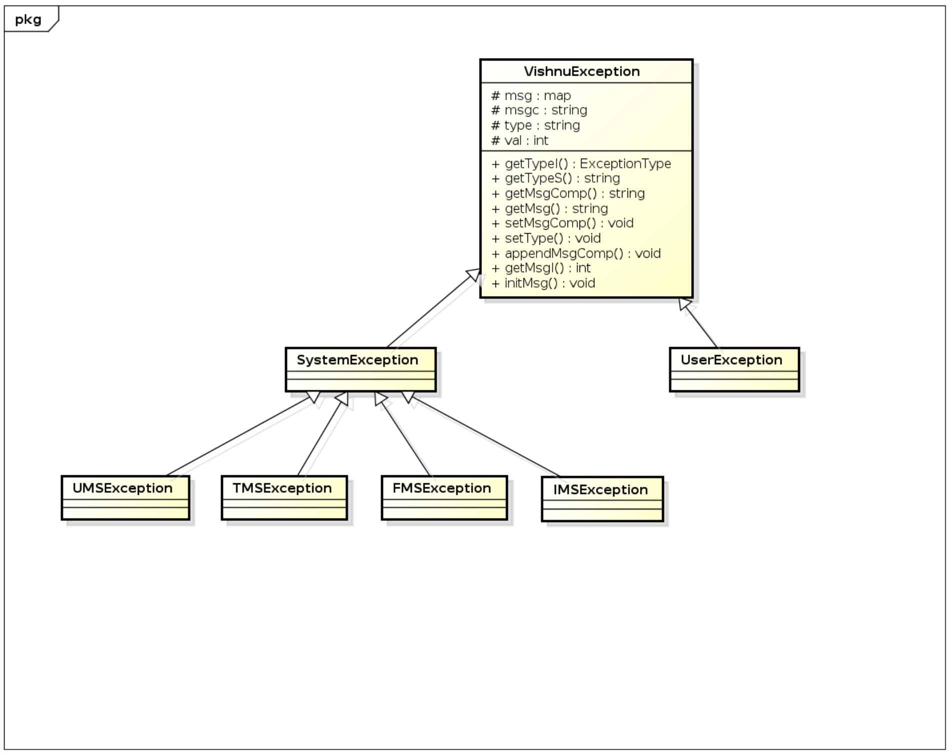

Figure 4.6: exception