|  |
| --- |
| Université paris-sud XI |
| ErLang IRC Serveur |
| Rapport d’Analyse |

|  |
| --- |
| Berthoux Vincent - Zermati Nicolas - Goncalves Vincent  03/03/2008 |

Table des matières

[Diagramme des cas d’utilisation : - 3 -](#_Toc192320603)

[Modèle des Objets : - 3 -](#_Toc192320604)

[Diagramme d’Objet : - 3 -](#_Toc192320605)

[Ivariants : - 3 -](#_Toc192320606)

[Client : - 3 -](#_Toc192320607)

[Serveur : - 3 -](#_Toc192320608)

[Canal : - 4 -](#_Toc192320609)

[Dictionnaire des Données : - 5 -](#_Toc192320610)

[IRC Message - 5 -](#_Toc192320611)

[Server Node - 5 -](#_Toc192320612)

[Client - 5 -](#_Toc192320613)

[Chan - 6 -](#_Toc192320614)

[Permissions - 6 -](#_Toc192320615)

[Perm Info - 6 -](#_Toc192320616)

[Modèle d’opération : - 7 -](#_Toc192320617)

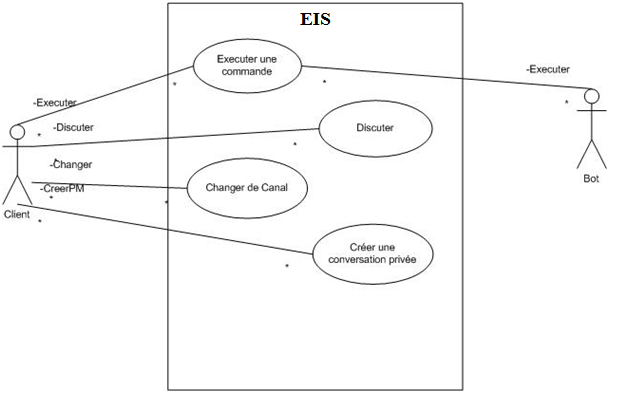
[Scénario d’utilisation : - 7 -](#_Toc192320618)

[Note : - 7 -](#_Toc192320619)

[Annexes - 8 -](#_Toc192320620)

[Annexe 1.A : Diagramme d’Objet : - 8 -](#_Toc192320621)

# Diagramme des cas d’utilisation :



# Modèle des Objets :

## Diagramme d’Objet :

Pour plus de clarté le diagramme d’Objet est disponible en annexe (cf. Annexe 1.A).

En revanche le langage de programmation choisit, étant le langage ErLang qui est un langage fonctionnel orienté processus. Le diagramme de classe présenté en annexe a été légèrement modifié pour pouvoir s’accorder à un diagramme de classe traditionnel.

## Ivariants :

Dans cette partie nous présentons les différents invariants liés aux clients, au serveur et aux canaux :

### Client :

1. Le pseudo (Nick) de l’utilisateur est unique :

context Client

inv : forAllInstances()->isUnique( Nick )

1. Un utilisateur ne peut reçevoir les messages d’un canal que s’il est présent sur ce canal.

### Serveur :

1. Seul les canaux visibles peuvent être listé par le client :

context ServerNode::List

post : result->forAll( c | NOT c.chanRights.isInvisible() )

### Canal :

1. Le nom du canal commence forcement par '#' ou '&' :

context Chan

inv : chanName->first() = '#' OR chanName->first() = '&'

1. Le nom de canal est unique :

context Chan

inv : forAllInstances->isUnique(chanName)

1. Le nombre d'utilisateur sur le canal ne peut excéder userLimit :

context Chan

inv : userLimit >= clients->size()

1. Un utilisateur banni ne peut rejoindre un chan :

context Chan::Join( user : CommonClient, sendedPassword : string )

inv : not isUserBaned( cli ) -- règle 4

1. Si le canal est en mode *voiced* alors il faut avoir le mode +v pour que les messages soient transmis aux autres utilisateurs :

context Chan::Privmsg( sender : CommonClient, message : Text )

inv : (chanRights.isModed() AND clients->select( c | c = sender AND c.rights.isVoiced() )) OR NOT chanRights.isModed()

1. Si le canal est protégé par mot de passe, seul un utilisateur fournissant le bon mot de passe pourra le rejoindre :

context Chan::Join( user : CommonClient, sendedPassword : string

inv : password = "" OR sendedPassword = pasword

1. La liste de clients *diffusés* ne peut pas être un client lointain :

context Chan

inv : broadcastedClients->forAll( c | no oclIsTypeof(RemoteClient))

1. Tous les clients d'un canal doivent avoir celui ci dans leur liste :

context Chan

inv : broadcastedClients->forAll( c | c.isPresentIn->exists( chan | chan = self() )

# Dictionnaire des Données :

## IRC Message

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Code** | **Nom** | **Légende** | **Type** | **Règles de gestion** |
| MSG\_SENDER | sender | Expéditeur du message | String | Obligatoire, non null |
| MSG\_RECEIVER | dest | Destinataire du message | String | Obligatoire, non null |
| MSG\_COMMAND | command | Commande obligatoire | String | Obligatoire, non null |
| MSG\_PARAMETERS | parameters | Paramètres optionnels de la commande | String | Facultatif |
| MSG\_DATA | data | Données optionnelles du message | String | Facultatif |

## Server Node

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Code** | **Nom** | **Légende** | **Type** | **Règles de gestion** |
| SRV\_PID | thisPid | Pid du processus serveur | Pid | Non null |

## Client

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Code** | **Nom** | **Légende** | **Type** | **Règles de gestion** |
| CLI\_NICK | nick | Pseudonyme du client | String | Unique, non null |
| CLI\_HOST | host | URI | String | Non null |
| CLI\_USERNAME | username | Nom réel de l'utilisateur | String | Non null |

## Chan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Code** | **Nom** | **Légende** | **Type** | **Règles de gestion** |
| CHN\_NAME | chanName | Le nom du canal | String | Non null, commence par un # ou un & |
| CHN\_LIMIT | userLimit | La limite d'utilisateurs présents | Entier | > 1 et < Limite fixée par le serveur |
| CHN\_PWD | password | Le mot de passe d'accès au canal | String | Optionnel, ne doit pas contenir de caractères spéciaux voir la commande mode |
| CHN\_BAN\_LIST | banList | La liste des utilisateurs banni | Liste de string |  |
| CHN\_TOPIC | topic | Le sujet du canal | String |  |

## Permissions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Code** | **Nom** | **Légende** | **Type** | **Règles de gestion** |
| PRM\_MODE | mode | Permissions pour un canal ou pour un utilisateur | Entier |  |

## Perm Info

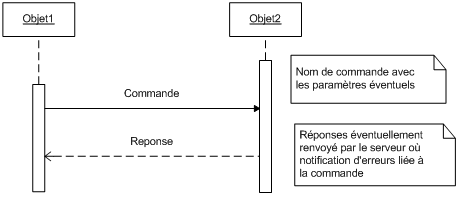
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Code** | **Nom** | **Légende** | **Type** | **Règles de gestion** |
| INF\_CLIENT | client | Pseudonyme du client dont on veut définir les permissions dans INF\_CHANNEL | String |  |
| INF\_PERM | permissions | Permissions dans INF\_CHANNEL d'un client | Permissions |  |
| INF\_CHANNEL | chan | Nom du canal dans lequel INF\_CLIENT a les permissions INF\_PERM | String |  |

# Modèle d’opération :

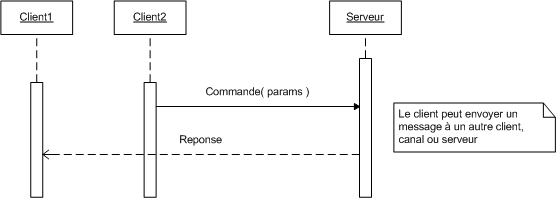
# Scénario d’utilisation :

Les scénarios présentés ci-dessous, montrent, de façon générale, le fonctionnement du serveur.

Le premier scénario montre l’exécution d’une commande simple entre le client et le serveur. Selon le type de la commande envoyé, le serveur renvoi ou non une réponse. Une réponse peut être le résultat de la commande envoyé ou alors une erreur propre à la commande.



Le deuxième scénario est utilisé lors d’échange de message entre les clients. Un premier client envoi son message au serveur. Enfin le serveur envoi le message au(x) destinataire(s) ou renvoi une erreur.



## Note :

Comme nous avons décidé avec notre client de ne faire que la partie serveur du projet, le manuel d’utilisateur ne peut être écrit en ce moment. Ce manuel contiendra les informations nécessaires à l’administrateur de se serveur pour la maintenance.

# Annexes

## Annexe 1.A : Diagramme d’Objet :

