Erlang IRC Server

Cahier des charges

Auteurs :

Vincent Berthoux

Vincent Goncalves

Nicolas Zermati

Table des matières

[Présentation 3](#_Toc190505928)

[Le sujet du projet 3](#_Toc190505929)

[Environnement 3](#_Toc190505930)

[Description du service 4](#_Toc190505931)

[Périmètre du projet 4](#_Toc190505932)

[Description générale du projet 4](#_Toc190505933)

[Limites du projet 4](#_Toc190505934)

[Fonctionnalités de base du serveur 4](#_Toc190505935)

[Permettre à des clients de rejoindre des canaux de discussion 4](#_Toc190505936)

[Permettre à ces mêmes clients de dialoguer en privé 4](#_Toc190505937)

[Autoriser un réseau de serveurs 4](#_Toc190505938)

[Fonctionnalités avancées du serveur 4](#_Toc190505939)

[Proposer la réservation de pseudonyme (NickServ) 4](#_Toc190505940)

[Proposer l’enregistrement de canaux (ChanServ) 4](#_Toc190505941)

[Limiter le SPAM (SPAMBot) 5](#_Toc190505942)

[Limiter le flood (FloodBot) 5](#_Toc190505943)

[Solutions proposées 6](#_Toc190505944)

[Fonctionnalités de base du serveur 6](#_Toc190505945)

[Fonctionnalités avancées du serveur 6](#_Toc190505946)

[Proposer la réservation de pseudonyme (NickServ) 6](#_Toc190505947)

[Proposer l’enregistrement de canaux (ChanServ) 6](#_Toc190505948)

[SPAMBot et FloodBot 6](#_Toc190505949)

[Modules 6](#_Toc190505950)

[Coûts de ces solutions 6](#_Toc190505951)

[Coût humain 6](#_Toc190505952)

[Coût temporel 7](#_Toc190505953)

[Installation et maintenance 7](#_Toc190505954)

[Prévisions de fiabilité 7](#_Toc190505955)

[Perspectives d’évolutions et ajouts de modules 7](#_Toc190505956)

[Documentation 8](#_Toc190505957)

# Présentation

## Le sujet du projet

La demande qui nous a été faite était de concevoir un logiciel de messagerie instantanée. Ce type de logiciel se compose de milliers de clients et d’un serveur. Le but principal est de créer un client de messagerie.

## Environnement

Nous savons que plusieurs équipes développent un client compatible avec le protocole IRC ou MSN. De plus il existe déjà de nombreux clients dédiés à ce protocole, directement téléchargeable sur Internet.

En revanche, l’offre de créer un serveur IRC est, quand à elle, beaucoup plus réduite. Nous nous proposons donc de créer, à la place de se client, un serveur.

# Description du service

## Périmètre du projet

### Description générale du projet

Ce projet se compose d’un serveur compatible avec les clients IRC actuels. Afin de rendre ce serveur le plus agréable possible nous allons proposer des fonctionnalités utilisateur dépassant le cadre du simple serveur.

### Limites du projet

Nous nous limiterons aux fonctionnalités propres d’un serveur IRC (compatible RFC 2813) il n’est pas question de faire d’autre interface d’administration que celle définie dans la spécification du protocole. Nous ajouterons malgré tout quelques fonctionnalités avancées décrite ci-dessous.

## Fonctionnalités de base du serveur

Nous avons besoin de certaines fonctionnalités qui sont celles de n’importe quel serveur IRC. Elles se résument dans les points suivants.

### Permettre à des clients de rejoindre des canaux de discussion

Le logiciel serveur ne sera pas axé sur les contacts d’une personne, donc le serveur ne gérera pas de liste de contacts mais une liste de canaux de discussion que tout client pourra rejoindre. Un client présent sur un canal pourra suivre la conversation du groupe. Le serveur permettra donc la création et la modération des canaux.

### Permettre à ces mêmes clients de dialoguer en privé

On souhaite quand même conserver une certaine intimité donc on rendra possible les messages privés. Ces messages ne seront pas transmis aux autres utilisateurs que ceux spécifiés par l’envoyeur. Il sera de la responsabilité du client d’ignorer les messages privés d’un autre client.

### Autoriser un réseau de serveurs

Lorsqu’un serveur propose des services intéressant à ses utilisateurs alors il devient populaire. Cette popularité n’est pas négligeable puisqu’elle amène différents soucis d’échelle. C'est-à-dire que nous devrons être capables de supporter un grand nombre de clients, de canaux et de messages simultanément.

## Fonctionnalités avancées du serveur

Bien conscient du fait qu’IRC soit assez minimaliste, et que les utilisateurs ont d’autres besoins, nous proposons d’implanter les quelques fonctionnalités suivantes.

### Proposer la réservation de pseudonyme (NickServ)

Sans autre ajout il n’est pas prévu de garantir l’identité d’un pseudonyme aux autres membres du réseau. Pour combler à ce manque, nous fournirons un robot qui sera présent sur le serveur en permanence fournissant aux utilisateurs un service de réservation de pseudonymes. Un utilisateur pourra donc enregistrer un seul pseudonyme et lui attribuer un mot de passe.

### Proposer l’enregistrement de canaux (ChanServ)

De la même façon qu’il est important de garantir l’identité des utilisateurs il est normal de pouvoir déclarer un canal comme étant le sien et d’y avoir le statut d’opérateur en permanence. Pour cela nous mettrons en place un autre robot qui aura tout droit sur les utilisateurs et qui sera en charge de garantir que seul le créateur et les gens qu’il aura autorisé seront capables de modérer le canal.

### Limiter le SPAM (SPAMBot)

Malheureusement il existe des robots dédié au SPAM qui abaissent la qualité des serveurs en envoyant dans tous les canaux des messages publicitaire ou autre. Nous proposeront donc un système d’anti-SPAM.

### Limiter le flood (FloodBot)

À cause de la nature de la communication par canaux, on se heurte à certains devoirs de lisibilité envers les clients. En effet on peut imaginer qu’un client intentionnellement ou non puisse diminuer la qualité de service que nous souhaitons offrir en parlant de façon intempestive. Nous proposerons donc un robot qui sera en charge de surveiller que chacun des clients ne nuise pas au canal sur lequel il se trouve de cette façon.

# Solutions proposées

Pour réaliser les fonctionnalités ci-dessus nous allons mettre en place certains mécanismes. Ils sont détaillés ci-dessous.

## Fonctionnalités de base du serveur

La base du serveur sera réalisé en Erlang un langage spécialisé pour ce type d’application. L’Erlang est un langage fonctionnel orienté processus. Néanmoins nous vous fournirons une analyse, une conception et une documentation toutes aussi complètes qu’avec un langage objet comme Java. Ce langage comportant de nombreux avantages de développement il semble s’imposer naturellement pour ce projet.

## Fonctionnalités avancées du serveur

La base du serveur sera de telle sorte que nous puissions lui ajouter des fonctionnalités aisément. Ces fonctionnalités sont présentées ci-dessous.

### Proposer la réservation de pseudonyme (NickServ)

Le NickServ sera un module du serveur, il viendra s’ajouter dans les utilisateurs du serveur et les messages lui étant adressés seront des commandes particulière de réservation. Il y aura certaines limites, par exemple un client ne pourra pas réserver un pseudonyme indéfiniment, ou bien une quantité illimitée de pseudonyme.

### Proposer l’enregistrement de canaux (ChanServ)

De même que pour le NickServ le ChanServ sera un module du serveur qui sera visible par les utilisateurs qui pourront lui adresser des messages de réservation. Le ChanServ aura certain quotas pour les utilisateurs qui limiteront le nombre de canaux qu’il pourrait réserver.

### SPAMBot et FloodBot

Le SPAM et le Flood seront quand à eux gérés par des robots, pas nécessairement différents, qui seront présents sur chaque canal. Un opérateur pourra demander aux administrateurs ou à un robot de lui envoyer ces robots sur son canal. Ils feront partie du module de gestion des canaux.

## Modules

Le but de notre conception sera de pouvoir par la suite ajouter sans aucuns problèmes d’autres fonctionnalités avancés avec le minimum de contraintes. Nous faciliterons la mise en place de robots du même type que le FloodBot et le SPAMBot.

## Coûts de ces solutions

Mettre en place ce genre de solution n’est pas gratuit. Nous avons des besoins humains et temporels que voici.

### Coût humain

Nous seront trois analystes, trois concepteurs et trois programmeurs. Peut être seront nous amené à faire appel à une aide extérieure lors de la conception ou du développement.

### Coût temporel

Nous disposons d’environs trois mois pour réaliser les phases d’analyse, de conception, de développement et de tests. À la fin de chacune de ces phases nous vous fournirons un rapport d’avancement.

## Installation et maintenance

Afin de pouvoir faire fonctionner ce programme, il faudra que la machine hôte soit en mesure de faire fonctionner la machine virtuelle Erlang. Ceci ne devrait pas poser de problème puisqu’Erlang est un langage open source. Il faudra également que la machine hôte dispose d’un accès très haut débit afin de pouvoir supporter beaucoup de connexion.

## Prévisions de fiabilité

Le déploiement d’une application serveur nécessite une fiabilité accrue. Pour cela nous misons beaucoup sur le langage que nous avons choisit qui diminue le nombre de lignes de code donc le nombre de bogues, et qui dispose de technique de gestion d’erreurs de qualité. L’application pourra donc se relancer elle-même en cas d’erreur et bénéficiera d’une haute disponibilité puisque le langage étant pseudo-interprété dans une machine virtuelle nous pouvons changer le code à chaud.

## Perspectives d’évolutions et ajouts de modules

Pour reprendre ce que nous avons dit sur les modules, nous envisageons de faire évoluer l’application assez facilement. Nous pourrions par exemple ajouter un robot de Quizz.

# Documentation

#### IRC

Vous pouvez trouver les références du protocole IRC à cette adresse :

<http://www.irchelp.org/irchelp/rfc/>

#### Erlang

Le site officiel d’Erlang est à cette adresse :

<http://www.erlang.com/>

#### EIS wiki

Le wiki de notre projet, ou nous échangeons nos informations de développement :

<http://217.70.188.170/trac/jim/>