



Centre d'études de populations, de pauvreté et de politiques socio-économiques/International  
Networks for Studies in Technology, Environnement, Alternatives, Development

## *Travail de fin d'étude :*

# Création et administration d'un serveur Syslog

Ecrit par :

**Coléry David**  
Rue du Calvaire, 2  
7911 Moustier  
Tel : +32 475 47 12 52  
Mail : [colery.david@skynet.be](mailto:colery.david@skynet.be)



HAUTE ECOLE LIBRE DU HAINAUT OCCIDENTAL  
**Département Technique :**  
**Informatique et systèmes – finalité :**  
**Technologie de l'informatique**  
Rue Frinoise, 12 – 7500 TOURNAI  
Tél : 069 89 05 60 – Fax : 069 89 05 65  
E-mail : [dep.technique@helho.be](mailto:dep.technique@helho.be)

**Année Scolaire 2004-2005.**



Je tiens à remercier Messieurs M. Leman et B. Clicque pour nous avoir supportés durant ces semaines des stages. Remercier aussi le CEPS / INSTEAD pour nous avoir permis de loger sur place, ainsi que toutes les personnes qui ont permis que ce rapport voie le jour.



# Tables des matières

Tables des matières .....	5
<b>Présentation de l'entreprise.....</b>	<b>7</b>
<b>Historique.....</b>	<b>7</b>
<b>Les missions du centre .....</b>	<b>7</b>
<b>Sa politique de recherche.....</b>	<b>8</b>
<b>Sa politique de développement.....</b>	<b>8</b>
<b>Son mode de fonctionnement .....</b>	<b>8</b>
<b>Ses principales études et recherches .....</b>	<b>9</b>
<b>1. Présentation du stage .....</b>	<b>10</b>
<b>Approche générale.....</b>	<b>10</b>
<b>Logiciels utilisés pour la réalisation du projet.....</b>	<b>10</b>
<b>2. L'apprentissage en Coldfusion.....</b>	<b>11</b>
<b>Structure du site .....</b>	<b>11</b>
<b>Structure de la base de données .....</b>	<b>12</b>
<b>3. Présentation du projet .....</b>	<b>13</b>
<b>Cahier des charges .....</b>	<b>13</b>
<b>Qu'est-ce qu'un Serveur Syslog ? .....</b>	<b>13</b>
<b>La norme RFC 3164.....</b>	<b>14</b>
<b>4. Le serveur Syslog.....</b>	<b>17</b>
<b>La classe Form1 .....</b>	<b>17</b>
Le contrôle MainMenu :.....	18
Le contrôle ListView :.....	19
Le contrôle Panel :.....	20
Le contrôle StatusBar :.....	20
<b>La classe CSyslogServer .....</b>	<b>20</b>
Un Socket :.....	21
La classe Socket : .....	21
La classe IPEndPoint :.....	22
<b>La classe CMyMessage .....</b>	<b>23</b>
La détection des messages : .....	23
La classe Regex :.....	23
<b>La classe CRfcSpecs .....</b>	<b>24</b>
<b>Le Formulaire Send .....</b>	<b>24</b>
<b>Le formulaire Connection .....</b>	<b>26</b>
La classe File :.....	26
La classe StreamWriter : .....	27
La classe StreamReader : .....	27
La classe SqlConnection : .....	28
Le cryptage :.....	29
<b>Le formulaire adresse .....</b>	<b>29</b>
<b>La base de données.....</b>	<b>30</b>
<b>Difficultés rencontrées .....</b>	<b>30</b>

<b>5. Le site d'administration</b> .....	31
<b>La page d'accueil</b> .....	31
<b>Tout voir</b> .....	31
<b>La Recherche</b> .....	31
Par severity.....	32
Par facility .....	32
Par hostname .....	32
Par date.....	32
Recherche avancée.....	32
<b>La purge</b> .....	33
<b>Les statistiques</b> .....	33
<b>Les difficultés rencontrées</b> .....	33
<b>6. Sql Server 2000</b> .....	34
<b>Ses bénéfices</b> .....	34
<b>Ses types de données</b> .....	34
<b>7. Conclusion</b> .....	35
<b>Bibliographie :</b> .....	36
▪ <b>Ouvrages Consultés</b> .....	36
▪ <b>Sites Consultés</b> .....	36

## **Présentation de l'entreprise**

Le CEPS/INSTEAD (Centre d'études de populations, de pauvreté et de politiques socio-économiques / International Networks for Studies in Technology, Environment, Alternatives, Development) est un centre et réseau de recherches socio-économiques. Il se trouve à Differdange au Grand-duché de Luxembourg, près des frontières belge et française. Il occupe une centaine de personnes.

### **Historique**

Le CEPS/INSTEAD a été créé en mars 78 par Gaston Schaber, professeur à l'université de Liège. La commission Européenne lui avait demandé de réaliser un projet comparatif transnational ayant pour thème : "Étude sur la pauvreté persistante dans sept régions de cinq pays industrialisés."

Pour réaliser ce projet, M. Schaber fonda avec un groupe de chercheurs et de travailleurs sociaux une association sans but lucratif, le Groupe d'Étude pour les Problèmes de la Pauvreté (GEPP), qui est donc l'ancêtre du CEPS, et établit un réseau permanent entre sept centres de recherche.

Depuis novembre 89, le CEPS/INSTEAD asbl est institué comme établissement public jouissant de l'autonomie scientifique, financière et administrative.

Le CEPS/INSTEAD est administré par un conseil d'administration composé de neuf membres, dont M. Schaber en est le président. Les membres sont nommés par le gouvernement.

### **Les missions du centre**

Le centre a pour missions :

- de faire et d'organiser des études ponctuelles et longitudinales de populations, de pauvreté, et de politiques socio-économiques ;
- de créer, de gérer, et d'exploiter des banques de données scientifiques nationales et internationales comparatives ;
- de développer des instruments d'analyse, de modélisation et de simulation pour politiques socio-économiques ;
- de développer et d'améliorer des outils informatiques pour les domaines socio-économiques ;
- de créer et d'entretenir des réseaux interrégionaux et internationaux de recherche et d'information en matière de technologies, d'environnement, de développement et de solutions alternatives de développement ;
- d'organiser au niveau post-gradué des formations en rapport avec les recherches envisagées.

## **Sa politique de recherche**

Aux différents niveaux, national, régional, interrégional et international, le CEPS/INSTEAD effectue des recherches micro-économiques et micro-sociales et crée des banques de données micro-économiques et micro-sociales, dans le but de développer des approches nouvelles pour l'analyse, la programmation et la simulation de politiques socio-économiques. Ces recherches et ces banques de données visent à produire des informations nouvelles, ajouter de la valeur à des données conventionnelles en créant de la compatibilité et de la comparabilité, développer des méthodologies innovatrices et des instruments utiles pour le monitoring de politiques ou le transfert d'informations.

## **Sa politique de développement**

Le CEPS/INSTEAD développe et consolide ses réseaux de chercheurs et de recherche à travers l'exécution commune de projets nationaux et transnationaux sous contrat. Actuellement les réseaux fonctionnent principalement en sciences économiques et sociales, mais les travaux s'élargissent progressivement vers les sciences exactes et les technologies, pour que les réseaux servent encore mieux au développement et au redéveloppement économique et social des régions.

Les études nationales sont conçues de façon à pouvoir intéresser d'autres pays pour leur valeur de prototype.

Les études internationales ou interrégionales ont parmi leurs premiers objectifs de produire de la comparabilité.

## **Son mode de fonctionnement**

Le centre fonctionne sur un mode

- interdisciplinaire, avec des collaborateurs permanents des disciplines suivantes : psychologie, pédagogie, sociologie, économie, économétrie, mathématiques, informatique, géographie et (plus récemment) sciences exactes et technologies,
- interuniversitaire, avec des collaborateurs réguliers des universités/instituts universitaires de recherche de Liège, Nancy, Anvers, Bruxelles, Tilburg, Strasbourg, Paris, Louvain, Leuven, Berlin (DIW), Mannheim, Ann Arbor (Michigan), Clark University, Harvard, M.I.T. (Massachusetts), Syracuse (New York), Pittsburgh (Pennsylvania) et, avec un contrat de coopération à long terme, avec la Florida Atlantic University, Boca Raton (1991), ainsi que des collaborateurs scientifiques des pays d'Europe Centrale (Université de Budapest, TARKI Budapest, Université de Varsovie, Université de Prague) des pays de l'ancienne Union Soviétique, plus particulièrement de la Russie (Russian Academy of Sciences (Moscou), Institute for the Economy in Transition (Moscou), Institute of Macroeconomic Research and Forecasting at the Higher School of Economics (Moscou)),
- interrégional : Nancy, Sarrebruck, Cologne, Limburg, Maastricht, Tilburg, Arlon, Metz, Longwy.

## Ses principales études et recherches

Le centre travaille en premier lieu sur base d'une Convention établie entre le Gouvernement et le CEPS/INSTEAD, et hors convention, sur demande, pour des ministères, des administrations et des organismes semi-publics ou privés. Il travaille également, sous contrat, pour de grandes fondations scientifiques internationales et pour des Directions Générales de la Commission Européenne.

### Études et recherches nationales

- Panel Socio-économique "Liewen zu Lëtzebuerg" (PSELL)
- Enquête Dynamique sur les entreprises au Luxembourg (EDEL)
- Formation, Education et Emploi (FEE)

### Études et recherches comparatives/internationales

- Luxembourg Income Study
- Luxembourg Employment Study
- PACO Ménages
- PACO Entreprises
- Nouveaux Développements

# **1. Présentation du stage**

## **Approche générale**

Durant les premières semaines, nous avons eu une phase d'apprentissage pendant laquelle nous avons étudié le *Coldfusion* et le *Visual C++*. Pour nous faciliter la tâche, nos maîtres de stages, Mr Clique et Mr Leman, nous ont demandé de réaliser un site qui montre notre état d'avancement.

Initialement, il y avait quatre projets pour ce stage, et après concertation avec mes collègues, nous avons chacun choisi le nôtre. Mon projet consiste à créer un programme en *Visual C++* qui capture tous les messages circulant sur le réseau et à les stocker dans une base de données. Une fois dans la base de données, l'administrateur du réseau peut visualiser ces messages dans un site écrit en *Coldfusion*. Ce site permettra l'affichage des messages de manière sélective c'est-à-dire que l'administrateur pourra choisir les messages qu'il veut visualiser. Ce projet a pour but de permettre à un administrateur réseau de visualiser les éventuels problèmes du réseau ou les éventuelles attaques de pirates.

## **Logiciels utilisés pour la réalisation du projet**

### ***Visual Studio.NET.***

*Visual Studio.NET* est un outil que nous avons appris à utiliser pendant notre scolarité. Cet outil nous permet de programmer dans divers langages dont le *Visual C++*. Les programmes écrits en langage *.Net* nécessitent le "*dotNet Framework*" pour fonctionner. Celui-ci est téléchargeable gratuitement sur le site de Windows Update.

### ***Dreamweaver MX***

J'ai utilisé ce logiciel de *Macromedia* comme éditeur de code *Html*, *Coldfusion*, parce qu'il offre de nombreux atouts comme la reconnaissance des balises, une aide assez bien fournie et facile d'accès et une interface agréable pour programmer.

### ***Coldfusion MX Server***

Ce logiciel nous permet d'interpréter du code *ColdFusion* enregistré dans des pages à l'extension "\*.cfm" et via un serveur web, de le mettre à disposition sur internet. Il nous permet d'aller interroger ou d'écrire dans la base de données pour afficher les documents voulus.

### ***SQL Serveur 2000***

Ce logiciel nous permet de gérer des bases de données de plus grande envergure qu'Access. Les bases de données en *SQL Serveur* sont bien plus performantes que celles d'Access qui est limité en vitesse d'exécution et en flexibilité. Le CEPS/INSTEAD possède un serveur de base de données *SQL Serveur* qui gère toutes leurs bases de données.

## **2. L'apprentissage en Coldfusion**

Comme il est stipulé dans le guide du travail de fin d'étude, pour notre stage, nous avons la consigne de tenir un cahier de stage, sorte de "journal de bord", dans lequel nous consignons jour après jour, nos activités durant le stage. Ce cahier nous permettra, si besoin en est, de justifier notre emploi du temps dans l'entreprise et peut aussi servir d'aide mémoire pour la rédaction de notre rapport. C'est exactement ce que le site représente mais de manière non officielle et juste au sein de l'entreprise.

Le site est donc réalisé en *Coldfusion*, langage apparenté à l'*HTML*, qui permet de réaliser des sites Internet ou Intranet de manière plus dynamique. Grâce au serveur *Coldfusion MX*, fourni par *Macromedia*, ce langage nous permet d'accéder à une base de données et ainsi créer un dynamisme dans les pages.

### **Structure du site**

Le site comprend une page d'index sur laquelle on retrouve un menu pour naviguer dans le site quand on est visiteur. Celle-ci fournit en plus la possibilité de se connecter (logger) au site si on y est inscrit et que l'on possède des droits autres que la visite (admin ou user).

Si on n'est que visiteur, le menu nous permet de visualiser la page d'accueil, la page de présentation des quatre étudiants, ainsi que la page d'affichage des résultats.

La page d'accueil contient un petit message de bienvenue et l'adresse du CEPS/INSTEAD. La page de présentation contient une petite présentation des quatre étudiants ayant réalisé le site et pour qui le site est utile. La page de visualisation permet de visualiser les commentaires sur les activités journalières des quatre étudiants en sélectionnant le jour, le mois et le nom de l'étudiant à l'aide des listes déroulantes.

Si on se connecte à l'aide de la zone prévue à cet effet, le menu de la page d'index affiche des nouvelles possibilités suivant que l'on soit utilisateur (user) ou administrateur (admin) sur le site. Les données concernant les droits d'accès sont contenues dans une table de la base de données.

Si on se connecte en tant qu'utilisateur (user), la possibilité de navigation est l'ajout et la modification de commentaires dans la base de données.

En tant qu'administrateur (admin), il existe la possibilité supplémentaire de gérer les utilisateurs.

Dans la page ajout et modification de commentaires, on peut sélectionner le jour et le mois et le nom d'utilisateur pour lequel on veut ajouter un commentaire. S'il existe déjà un commentaire, nous sommes dirigés vers une page qui l'affiche et nous permet de le modifier grâce à une requête UPDATE. Si par contre il n'y a pas encore de commentaire, nous sommes dirigés vers une page avec un zone de texte nous permettant de taper notre commentaire et de l'insérer grâce à une requête INSERT.

Dans la page gestion des utilisateurs, on peut visualiser les différents utilisateurs présents dans la base de données et on peut ajouter ou supprimer un utilisateur grâce au formulaire prévu à cet effet.

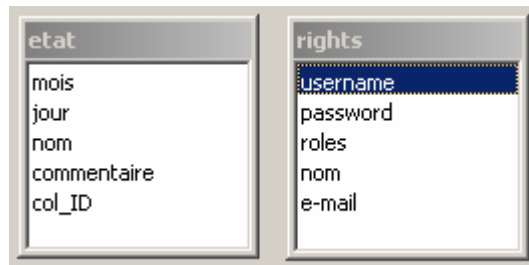
## Structure de la base de données

Une table "etat" qui contient cinq champs :

- "mois" de type entier
- "jour" de type entier
- "nom" d'une taille de 50 caractères
- "commentaire" d'une taille de 1000 caractères
- Col\_ID de type entier ; c'est un numéro automatique qui sert de clé primaire

Une table "rights" qui contient cinq champs :

- "username" d'une taille de 50 caractères
- "password" d'une taille de 50 caractères
- "roles" d'une taille de 50 caractères
- "nom" d'une taille de 50 caractères
- "e-mail" d'une taille de 255 caractères



The image shows two side-by-side windows representing database tables. The left window is titled 'etat' and lists five fields: 'mois', 'jour', 'nom', 'commentaire', and 'col\_ID'. The right window is titled 'rights' and lists five fields: 'username', 'password', 'roles', 'nom', and 'e-mail'. The 'username' field in the 'rights' table is highlighted with a blue selection bar.

Table Name	Field Name
etat	mois
	jour
	nom
	commentaire
	col_ID
rights	username
	password
	roles
	nom
	e-mail

**Fig 1 : Structure de la base de données du site.**

## **3. Présentation du projet**

### **Cahier des charges**

Mon projet consiste à créer un "Serveur Syslog" et à gérer son administration. Pourquoi créer ce programme alors qu'il en existe en vente ou même gratuit? Et bien c'est très simple, parce qu'avec un programme que nous développons nous même, nous avons plusieurs avantages, comme :

- La connaissance et l'accès au code source du programme pour une meilleure gestion.
- La possibilité de corriger les problèmes éventuels qui pourraient survenir,
- La gestion et le stockage des messages de la manière la plus appropriée pour l'administrateur du réseau.
- La possibilité de faire évoluer le programme au cours du temps et suivant les besoins de l'entreprise.

Pour être clair, je dois créer un programme en *Visual C++* qui capture tous les messages qui transitent sur le réseau. Le stockage de ces messages se fera dans une base de données qui sera gérée par *SQL Server 2000*. Une fois cela effectué, je dois réaliser un site intranet en *Coldfusion* qui permettra l'administration et l'affichage de ces messages.

### **Qu'est-ce qu'un Serveur Syslog ?**

Un *serveur Syslog* est un programme qui capte tous les messages "syslog" qui sont des messages d'événements générés par toutes les machines du réseau que ce soit des ordinateurs ou des routeurs ou les périphériques connectés sur le réseau.

Normalement, les messages sont écrits dans des fichiers de log, sorte de fichiers journaux, informant l'administrateur de ce qui se passe dans son réseau ou sur sa machine, et portant l'extension "\*.log".

Les messages *syslog* possèdent une structure commandée par la norme "RFC 3164" qui a été définie pour assurer une uniformité pour tous les constructeurs de matériel.

## La norme RFC 3164<sup>1</sup>

Cette norme décrit le comportement du protocole syslog. Ce protocole est utilisé depuis plusieurs années pour la transmission des messages d'événements à travers un réseau. Il a été développé par l'université de Californie Berkeley Software Distribution (BSD) et ensuite intégré dans les différents systèmes d'exploitation et dispositifs de réseau.

En des termes plus simplistes, le protocole syslog fournit une voie de transport pour permettre à une machine d'envoyer des messages d'événements, à travers un réseau d'adressage IP, aux collecteurs de messages d'événements - également connus sous le nom de serveurs de syslog. Le serveur syslog utilise un autre protocole, le protocole UDP (User Data Protocol – RFC 768), en tant que mécanisme fondamental de la couche de transport.

Le port UDP assigné au Syslog est le port 514, il est recommandé que le port de réception et le port d'envoi soient les mêmes, pour signaler correctement qu'il s'agit d'un message de type syslog.

La structure des messages, suivant la norme, est divisée en trois parties.

- La première partie est appelée "PRI",
- La seconde partie est "l'en-tête",
- La troisième partie est le "message". La longueur totale de ces messages n'excèdera pas 1024 bytes.

---

<sup>1</sup> Voir Annexes

- La partie "PRI" est composée de trois à cinq caractères. Elle commence avec le signe "plus petit (<)" suivi par un nombre et le signe "plus grand (>)" pour terminer. Le nombre situé entre les deux symboles correspond à la valeur de la priorité du message, cette priorité représente les codes de "severity" et de "facility" décrit plus bas. La valeur de la priorité consiste en un, deux ou trois chiffres décimaux [0,9]. La priorité est calculée en multipliant le code numérique de la "facility" par huit puis en ajoutant le code de la "severity". Si la priorité n'existe pas dans le message reçu, le serveur se charge lui-même de rajouter un priorité de valeur 13. (*facility*=1, *severity*=5).

Code numérique	Facility
0	Kernel Message
1	User-Level Messages
2	Mail System
3	System Daemon
4	Security/authorization messages (note 1)
5	Message generated internally by syslogd
6	Line printer subsystem
7	Network news subsystem
8	UUCP subsystem
9	Clock daemon (note 2)
10	Security/authorization messages (note 1)
11	FTP Daemon
12	NTP Subsystem
13	Log audit (note 1)
14	Log alert (note 1)
15	Clock daemon (note 2)
16	Local use 0 (local 0)
17	Local use 1 (local 1)
18	Local use 2 (local 2)
19	Local use 3 (local 3)
20	Local use 4 (local 4)
21	Local use 5 (local 5)
22	Local use 6 (local 6)
23	Local use 7 (local 7)

**Tableau 1 : Syslog messages facilities**

Code numérique	Severity
0	Emergency : system is unusable
1	Alert : Action must be taken immediatly
2	Critical : critical conditions
3	Error : Error conditions
4	Warning : warning conditions
5	Notice : normal but significant condition
6	Informational : informational messages
7	Debug : debug-level messages

**Tableau 2 : Syslog messages severities**

- La partie "En-tête" contient une indication de temps et le nom d'hôte ou l'adresse IP de la machine qui envoie le message. L'en-tête commence par l'indication de temps directement après le signe ">" de la partie PRI et les deux parties de l'en-tête sont séparées par un espace. Le champ du nom d'hôte correspond au nom que l'administrateur réseau a donné à la machine. Si il n'existe pas de nom d'hôte, alors ce champ correspondra à l'adresse IP de la machine. Le champ d'indication du temps correspond à la date et l'heure à laquelle le message a été envoyé, au format "Mmm dd hh:mm:ss" où :
  - Mmm correspond à l'abréviation anglaise du mois de l'année avec la première lettre en majuscule.
  - dd est le jour du mois, en deux chiffres sauf s'il est inférieur à 10.
  - Hh:mm:ss est l'heure locale avec l'heure représentée au format 24 heures, et comprise entre 00 et 23, et les minutes et les secondes comprises entre 00 et 59.

Si cette partie n'existe pas lors de la réception du paquet, alors l'appareil qui reçoit le message se charge de l'ajouter lui-même avec les infos qu'il possède sur la machine qui lui envoie le paquet.

- La partie "Message" remplit le reste du paquet *syslog*. Elle contient quelques indications supplémentaires sur le processus qui a généré le message et le texte du message. Ces indications sont séparées en deux blocs le "TAG Field" et le "CONTENT Field". Le TAG correspond au nom du processus qui a généré le message. Le CONTENT correspond aux détails du message.

## 4. Le serveur Syslog

Le programme du serveur *syslog* est donc développé en Visual C++, sous forme d'une application Windows, comme il a été dit précédemment. Il est composé des classes suivantes :

- Le formulaire *Form1* : La classe principale sous forme de formulaire.
- *CSyslogServer* : Classe contenant l'initialisation du *socket* d'écoute et de réception.
- *CRfcSpecs* : Classe contenant l'initialisation des codes de *facility* et de *severity* comme précisé dans la norme *syslog*.
- *CMyMessage* : Classe contenant la gestion des messages.
- Le formulaire *Send* : Fenêtre de dialogue qui simule l'envoi d'un message.
- Le formulaire *Connection* : Fenêtre de dialogue qui permet de créer un fichier texte qui contient les infos de connexion à la base de données.
- Le formulaire *Adresse* : Fenêtre de dialogue qui permet de créer un fichier texte qui contiendra l'adresse du site d'administration des messages.

### La classe **Form1**

Cette classe est composée de deux fichiers, le fichier "*Form1.h*" et le "*Form1.cpp*". Cette classe permet de créer la fenêtre de commande du programme (fig. 2).



Fig 2 : Fenêtre principale du programme

Cette fenêtre est composée de plusieurs contrôles :

- Le contrôle *MainMenu* contenant le menu de l'application.
- Le contrôle *ListView* permettant de visualiser les messages réceptionnés.
- Un contrôle *Panel* pour la légende.
- Une *StatusBar* qui montre le statut de l'application.

### Le contrôle MainMenu :

Comme son nom l'indique, ce contrôle représente le menu principal de l'application. Il est composé de "MenuItem", qui sont des objets qui créent la structure du menu.

Structure menu :

- Le menu Fichier

Composé de trois sous menus :

- Start server : qui permet de démarrer le serveur pour qu'il écoute et capte tous les messages qui lui arrivent. Ce sous menu est coché et donc activé au démarrage de l'application.
- Stop server : qui permet de stopper le serveur.
- Quitter : qui permet de quitter l'application.



**Fig 3 : Menu Fichier**

- Le menu Affichage

Composé de trois sous menus

- Afficher légende : qui permet d'afficher ou non le panel comprenant la légende.
- Clear liste : qui permet de nettoyer le contrôle *ListView* pour alléger l'affichage.
- View administration : qui permet d'ouvrir le site d'administration.

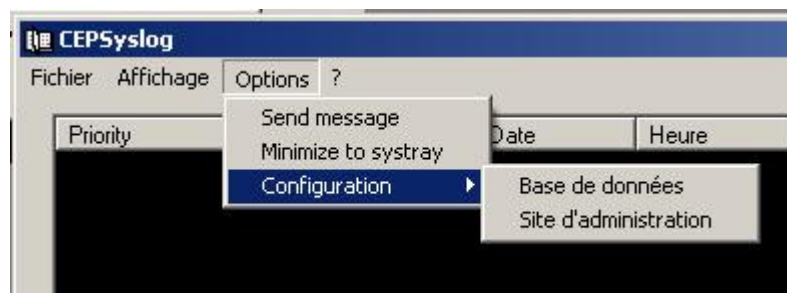


**Fig 4 : Menu Affichage**

- Le menu Options

Composé de trois sous menus :

- Send a message : sous-menu actif seulement si le serveur est démarré, et qui permet d'ouvrir le boîte de dialogue pour envoyer un message de test au serveur syslog.
- Minimize systray : qui permet de cacher l'application et de ne voir qu'une icône dans la systray (près de l'horloge).
- Configuration : Composé lui-même de deux sous menus :
  - Connection : qui permet d'ouvrir le formulaire Connection pour modifier les informations de connexion à la base de données.
  - Adresse du site : qui permet de modifier l'adresse du site d'administration

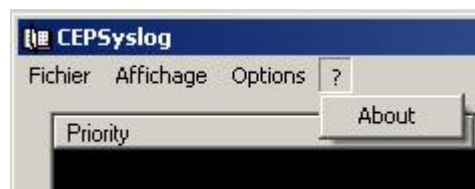


**Fig 5 : Menu Options**

- Le menu Help

Composé d'un sous menus :

- About : qui ouvre une boîte de dialogue qui donne des informations sur l'application.



**Fig 6 : Menu ?**

Le contrôle ListView :

Contrôle pouvant afficher une collection d'éléments de quatre façons différentes :

- LargeIcon : Grandes icônes
- SmallIcon : Petites icônes
- List : petites icônes dans une liste verticale.
- Détails : éléments affichés avec une série de sous éléments comportant des informations relatives à l'élément parent.

Ce contrôle fonctionne comme l'explorateur de Windows.

Pour ce projet, j'ai préféré le quatrième mode d'affichage, la vue détails qui permet d'afficher l'élément et ses sous éléments dans une grille munie d'en-têtes de colonnes qui identifie les données affichées pour les sous éléments.

### Le contrôle Panel :

Un Panel est un contrôle qui permet d'afficher les composants qu'il contient. Vous pouvez utiliser un panel pour l'une des fonctions suivantes :

- Regroupement des contrôles de manière logique de sorte qu'ils puissent être aisément affichés ou masqués.
- Définition d'un conteneur pratique où les contrôles peuvent être créés ou supprimés de manière dynamique.
- Utilisation d'un point unique pour appliquer des attributs de style à un jeu de contrôles, en les définissant sur le panel. Étant donné que l'héritage de style s'applique aux panels, les attributs définis sur un panel peuvent être hérités par des contrôles contenus dans ce panel.

### Le contrôle StatusBar :

Un contrôle *StatusBar* est constitué d'objets *StatusBarPanels* dont chacun affiche du texte et/ou une icône. Généralement, un contrôle *StatusBar* affiche des informations sur les objets affichés dans un Form, les composants de cet objet ou les informations contextuelles relatives au fonctionnement de cet objet au sein de votre application.

Dans ce projet, ce contrôle comporte cinq panels pour afficher des informations.

- Le statut du thread qui gère les actions du serveur syslog.
- Le statut du socket qui permet la communication réseau du serveur.
- Un compteur des messages reçus.
- L'heure système mise à jour toutes les secondes.
- La date du jour.

## **La classe CSyslogServer**

Cette classe est composée d'un fichier "*SyslogServer.h*" qui contient les déclarations des méthodes (fonctions) et des membres (variables) de la classe, et un fichier "*SyslogServer.cpp*" qui contient les initialisations des différents membres et les définitions des méthodes.

Elle a pour rôle de créer un *Socket* qui écoute le port UDP 514 pour capter l'arrivée des messages. Elle convertit ces messages en données lisibles par le programme et les envoie vers une autre classe pour leur traitement.

### Un Socket :

Un *Socket* est une extrémité de communication, c'est-à-dire un objet par lequel une application Windows Sockets envoie ou reçoit des paquets de données sur un réseau. Actuellement, les *Sockets* échangent en général des données uniquement avec d'autres *sockets* dans le même "domaine de communication", qui utilise Internet Protocol Suite.

Les deux types de *sockets* sont bidirectionnels ; il s'agit de flux de données qui peuvent être communiqués simultanément dans les deux sens. Vous disposez deux types de *sockets* :

- Socket flux : les sockets de flux assurent un flux de données sans limites d'enregistrement.
- Socket datagrammes : les sockets datagrammes prennent en charge un flux de données orienté enregistrement dont la livraison n'est pas garantie et qui risque de n'être ni séquencé comme lors de l'envoi, ni non dupliqué.

"Séquencé" signifie que les paquets sont remis dans le même ordre que lors de l'envoi.

"Non dupliqué" indique que vous ne recevez qu'une seule fois un paquet particulier.

Pour ce programme, comme le protocole *syslog* stipule que le serveur doit écouter le port UDP 514, le socket doit être un socket datagramme car :

Les *sockets datagrammes* sont "sans connexion" c'est-à-dire qu'aucune connexion explicite n'est établie. Un atout supplémentaire des *datagrammes* est la diffusion de messages à un grand nombre d'adresses réseau. Les *datagrammes* sont mieux adaptés aux données orientées enregistrement que les *sockets flux*.

Pour créer ce *socket*, il faut instancier un objet de la classe *Socket*.

### La classe Socket :

La classe *Socket* fournit une grande variété de méthodes et de propriétés pour les communications réseau. Elle vous permet d'effectuer des transferts de données synchrones et asynchrones à l'aide de n'importe quel protocole de communication.

Si vous utilisez un protocole sans connexion, tel que UDP, vous n'avez pas besoin d'écouter les connexions.

- En mode synchrone, appelez la méthode *ReceiveFrom* pour accepter tous les datagrammes entrant. Utilisez la méthode *SendTo* pour envoyer des datagrammes à un hôte distant.
- En mode asynchrones : utilisez les méthodes *BeginSendTo* et *EndSendTo* pour envoyer des datagrammes et les méthodes *BeginReceiveFrom* et *EndReceiveFrom* pour recevoir des datagrammes.

Lorsque vous avez terminé d'envoyer et de recevoir des données, utilisez la méthode *Shutdown* pour désactiver le socket. Après avoir appelé la méthode *Shutdown*, appelez la méthode *Close* pour libérer toutes les ressources associées au socket.

Pour créer cet objet de la classe et le connecter pour qu'il écoute un port spécifique,

```
1 Socket* ListenSocket;
2 IPEndPoint* LocalEndPoint;
3 String* dataRec=NULL;
4 //Création du socket d'écoute
5 this->ListenSocket = new Socket(this->LocalEndPoint->AddressFamily,
6     SocketType::Dgram, ProtocolType::Udp);
7 this->ListenSocket->Bind(this->LocalEndPoint);
8
9 //Recupération de l'hôte qui envoie le message
10 IPEndPoint* sender = new IPEndPoint(IPAddress::Any,0);
11 EndPoint* ep = __try_cast<EndPoint*>(sender);
12     while(true)
13     {
14         Byte buffer[]= new Byte[1024];
15         //Reception du message
16         int byteRec = this->ListenSocket->ReceiveFrom(buffer,&ep);
17         Encoding* ascii=Encoding::ASCII;
18         dataRec=ascii->GetString(buffer,0,byteRec);
19         //on souffle un peu pour que le message soit réceptionné complètement
20         System::Threading::Thread::Sleep(
21             System::TimeSpan::FromMilliseconds(10));
22         //On envoie le message dans la classe MyMessage pour qu'il soit traité
23         message->write(dataRec);
24     }
```

A la ligne 7, la méthode *Bind* associe le socket à un point de terminaison local (*IPEndPoint*). A la ligne 16, la méthode *ReceiveFrom(Byte buffer[],EndPoint\*\* ep)* lit les données reçues dans *buffer*, retourne le nombre d'octets lus correctement et capture l'adresse de l'hôte distant à partir duquel ont été envoyées les données. Cette méthode est particulièrement utile si vous avez l'intention de recevoir des datagrammes sans connexion d'un hôte inconnu ou de plusieurs hôtes.

La classe *EndPoint* est une classe abstraite qui identifie une adresse réseau.

#### La classe IPEndPoint :

Cette classe contient les informations relatives à l'hôte et au port dont l'application a besoin pour se connecter à un service d'un hôte. En combinant l'adresse IP de l'hôte et le numéro du port d'un service, la classe *IPEndPoint* crée un point de connexion à un service.

```
1 //Résolution du nom d'hôte du serveur grâce au DNS
2 IPHostEntry * hostInfo=Dns::Resolve(Dns::GetHostName());
3 //Récupération de l'adresse IP du serveur dans la liste correspondante
4 IPAddress *IPadresses = IPAddress::Parse(hostInfo->AddressList->
5     GetValue(0)->ToString());
6 LocalEndPoint=new IPEndPoint(IPadresses,514);
```

A la ligne 2, on retrouve la classe *IPHostEntry* qui fournit les informations sur l'adresse de l'hôte internet. Elle associe un nom d'hôte DNS (Domain Name System) à un tableau d'alias et à un tableau d'adresse IP correspondante. Elle joue aussi le rôle de classe d'assistance avec la classe *Dns* qui fournit les fonctionnalités de résolution de nom simple. A la ligne 4, on retrouve la classe *IPAdress* qui fournit une adresse IP.

## La classe CMyMessage

La classe *CMyMessage* gère la détection des messages et le transfert de ces messages vers la base de données et le *ListView* du formulaire Form1.

### La détection des messages :

Le paquet *syslog* qui est transmis au serveur doit être analysé pour vérifier que le format corresponde à celui énoncé par la norme *syslog*. Pour cela, j'ai décidé d'utiliser les expressions régulières. Pour utiliser ce type d'expressions, il existe en Visual C++ une classe qui s'appelle *Regex*.

### La classe Regex :

Cette classe représente une expression régulière<sup>2</sup> immuable.

- public : `Regex ( String* pattern);`  
*pattern* : modèle d'expression régulière à mettre en correspondance.

- public : `bool IsMatch (String* input);`  
*input* : chaîne dans laquelle une correspondance doit être recherchée.

- public `String* Replace(String* input, String* replacement);`  
*input* : chaîne à modifier  
*replacement* : chaîne de remplacement.

Exemple de pattern :

`"^\\<[0-9]{1,3}\\>"` : Cette expression associée avec la classe *regex* cherche une correspondance avec la partie PRI du paquet *syslog*.

Le symbole '^' est là pour définir que la correspondance doit être en début de chaîne.

[0-9] Chiffre de 0 à 9.

{1,3} de 1 à 3 fois.

Donc, l'expression suivante doit rechercher une correspondance avec la chaîne <nombre de 1 à 3 chiffres de 0 à 9>

```
"^((Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)\\s*(3[0-1]|2[0-9]|1[0-9]|[0-9]))\\s?(([0-1]\\d?|2[0-3]):([0-5]\\d?):([0-5]\\d?))"
```

Cette expression recherche une correspondance avec la partie d'indication de temps de la partie en-tête du paquet *syslog*. Elle doit donc rechercher une correspondance avec la date et l'heure d'envoi du message.

```
"^(25[0-5]|2[0-4]\\d|[0-1]?\\d?\\d)(\\. (25[0-5]|2[0-4]\\d|[0-1]?\\d?\\d)){3}"
```

Cette expression recherche une correspondance avec tout ce qui est ressemblant avec une adresse IP. Cette adresse IP est au format IPv4, c'est-à-dire *xxx.xxx.xxx.xxx* où x est un nombre de 0 à 255.

---

<sup>2</sup> Voir Annexes

## La classe CRfcSpecs

Cette classe implémente deux tableaux qui contiennent les codes de "facility" et de "severity" tel qu'ils sont définis dans la norme RFC 3164<sup>3</sup>  
<http://www.faqs.org/rfcs/rfc3164.html>.

Cette classe contient aussi des fonctions qui retournent les chaînes de caractères en correspondance avec le code de "severity" ou de "facility", et aussi des fonctions qui retournent le nombre d'éléments de ces deux tableaux.

## Le Formulaire Send

Cette fenêtre de dialogue sert à envoyer un message au serveur. Cette fenêtre est composée de deux Textbox, deux listes déroulantes, un label et deux boutons. Cette fenêtre est appelée par la méthode suivante :

```
1 System::Void m_send_Click(System::Object* sender, System::EventArgs* e)
2 {
3     //Creation d'un objet de la fenetre Send
4     Syslog::Send* envoi=new Syslog::Send();
5     //Affichage de la fenetre et recuperation du resultat du dialogue
6     if(envoi->ShowDialog()==DialogResult::OK)//si le resultat est ok
7     {
8         //récupération des données inscrite dans la fenetre de dialogue
9         String* host=envoi->txt_host->Text;
10        String* prio=envoi->labell1->Text;
11        String* message=envoi->txt_msg->Text;
12        String* processus="CEPSyslogGen";
13        String* num=compteur.ToString();
14        String* code=S"CEPSyslog";
15        //Envoi des données vers le socket
16        _syslogServer->SendMessage ( prio, host, num, code, processus,
17        message);
18        //rafraichissement des donnees du listview
19        this->dataReceived->Refresh();
20    }
```

A la ligne 4, on crée l'objet qui correspond à la boîte de dialogue.

A la ligne 6, on retrouve la fonction *ShowDialog()* qui permet d'appeler et afficher la boîte de dialogue.

---

<sup>3</sup> Voir Annexes

Grâce à DialogResult::OK et au test if on peut tester si le résultat renvoyé par la boîte de dialogue correspond à OK. Si oui, alors on effectue le code qui suit.



Fig 7 : Fenetre Send

La première *TextBox* correspond au nom d'hôte de la destination, mais comme cette fenêtre est interne au serveur le nom d'hôte correspondra donc au nom d'hôte du serveur.

La seconde *TextBox* correspond au message que l'utilisateur doit rentrer.

Les deux listes déroulantes correspondent respectivement aux chaînes de caractères des codes de "facility" et de "severity".

Lorsque l'on change les valeurs des listes déroulantes, le code de priorité est calculé directement et affiché dans le label prévu à cet effet.

Les boutons sont là pour le dialogue entre cette boîte de dialogue et la fenêtre principale *Form1*.

```
1 //calcul de la priorité suivant le code de severity
2 System::Void cb_severity_SelectedIndexChanged(System::Object* sender,
System::EventArgs* e)
3 {
4 //Calcul de la priorité lors du changement de sélection de la liste
déroulante
5     int i=this->cb_facility->SelectedIndex;
6     int j=this->cb_severity->SelectedIndex;
7     int res=(i*8)+j;
8     //Ecriture du calcul dans un label
9     this->labell1->Text=res.ToString();
10 }
```

## Le formulaire Connection

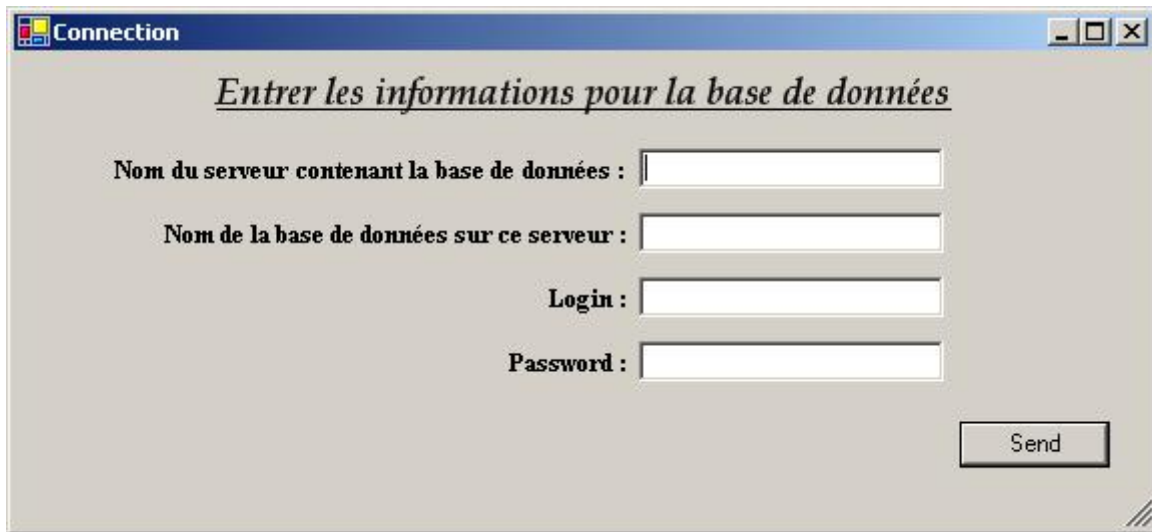


Fig 8 : Formulaire Connection

Ce formulaire permet de rentrer les informations nécessaires pour connecter le programme à la base de données SQL Server et de les stocker dans un fichier texte appelé "*MyConn.ini*".

Au démarrage, le programme vérifie si ce fichier est existant ou non, si il ne le trouve pas cette fenêtre apparaît avant la principale pour permettre à l'utilisateur de créer le fichier avec les données dont le programme a besoin.

Les informations que ce formulaire réclame sont des informations qui peuvent varier suivant le propriétaire du programme, c'est-à-dire :

- le nom du serveur de base de données qui contient la base de données que le programme va devoir utiliser. Le nom du serveur est en fait l'adresse IP de l'ordinateur servant de serveur.
- Le nom de la base de données qui se trouve sur le serveur.
- Le nom et le mot de passe du compte qui a un accès à cette base de données. Pour des raisons de sécurité, le mot de passe est crypté.

Pour stocker les informations dans un fichier, on utilise les classes File, StreamWriter et StreamReader. Pour connaître les informations à stocker il est aussi bon de connaître la classe SqlConnection.

### La classe File :

La classe File fournit les méthodes nécessaires pour la manipulation des fichiers.

La méthode la plus utilisée dans ce programme est la méthode Exists. Cette méthode, comme son nom l'indique, permet de vérifier l'existence d'un fichier.

```
public: static bool Exists(String* path);
```

*path* : Fichier à contrôler.

La méthode CreateText, qui permet de créer un fichier et de l'ouvrir en mode d'écriture:

```
public: static StreamWriter* CreateText(String* path);
```

La méthode `OpenText`, qui permet d'ouvrir le fichier en lecture :

```
public: static StreamReader* OpenText(String* path);
```

#### La classe `StreamWriter` :

Cette classe permet d'écrire des données sur le flux.

```
StreamWriter * sw = File::CreateText(path);
```

Cette ligne de code permet d'ouvrir un fichier dont le chemin est "path" en écriture. Pour écrire dans ce fichier, il faut faire appel à la méthode `Write` ou `WriteLine` suivant que l'on veuille écrire en continu ou ligne par ligne.

```
public: void Write(String* value);  
public: virtual void WriteLine(String* value);
```

#### La classe `StreamReader` :

Cette classe permet de lire des données sur le flux.

```
StreamReader* sr = File::OpenText(path);
```

Cette ligne de code permet d'ouvrir un fichier dont le chemin est "path" en lecture. Pour lire dans ce fichier, il faut faire appel à la méthode `Read` ou `ReadLine`.

```
public: int Read(\_\_wchar\_t buffer \_\_gc[], int index, int count);
```

Où les variables

*buffer* :

Lorsque cette méthode est retournée, la variable `buffer` contient le tableau de caractères spécifié dont les valeurs comprises entre *index* et (*index + count - 1*) sont remplacées par les caractères lus dans la source en cours.

*index* :

Index de *buffer* à partir duquel commencer l'écriture.

*count* :

Nombre maximal de caractères à lire.

```
public: String* ReadLine();
```

Cette méthode lit le texte contenu dans le fichier jusqu'au moment où elle découvre le caractère de nouvelle ligne, c'est-à-dire qu'elle lit le fichier ligne après ligne.

### La classe SqlConnection :

Représente une connexion ouverte à une base de données SQL Server. Un objet *SqlConnection* représente une session unique vers une source de données SQL Server. Dans le cas d'un système de base de données client/serveur, il équivaut à une connexion réseau au serveur.

Lorsque vous créez une instance de *SqlConnection*, les valeurs initiales sont affectées à toutes les propriétés. Pour modifier ces valeurs par défaut, il faut créer une chaîne de connexion que l'on associe à l'objet *SqlConnection* à l'aide de sa propriété *ConnectionString*.

Les paramètres de la chaîne de connexion sont :

Nom	Valeur par défaut	Description
Connection Timeout	15	Durée d'attente (en seconde) préalable à l'établissement d'une connexion au serveur avant que la tentative soit abandonnée et qu'une erreur ne soit générée.
Server		Nom ou adresse réseau de l'instance de SQL Server à laquelle se connecter.
Database		Nom de la base de données
Integrated Security	'false'	Lorsque la valeur est <b>false</b> , l'ID d'utilisateur et le mot de passe sont spécifiés dans la connexion. Lorsque la valeur est <b>true</b> , les informations actuelles d'identification du compte Windows sont utilisées pour l'authentification.  Les valeurs reconnues sont <b>true, false, yes, no</b> et <b>sspi</b> (vivement recommandée), qui équivaut à <b>true</b> .
Password		Mot de passe de la session du compte SQL Server (non recommandé. Pour garantir le plus haut niveau de sécurité, Il est vivement recommandé d'utiliser de préférence le mot clé Integrated Security ou Trusted_Connection).
Persist Security Info	'false'	Lorsqu'elles ont comme valeur <b>false</b> ou <b>no</b> (vivement recommandée), les informations de sécurité, comme le mot de passe, ne sont pas retournées dans le cadre de la connexion si celle-ci est ouverte ou l'a été à un moment donné. Le rétablissement de la chaîne de connexion rétablit toutes les valeurs des chaînes de connexion, y compris le mot de passe. Les valeurs reconnues sont <b>true, false, yes</b> et <b>no</b> .
User ID		Compte de connexion SQL Server (non recommandé. Pour garantir le plus haut niveau de sécurité, Il est vivement recommandé d'utiliser de préférence le mot clé Integrated Security ou Trusted_Connection).

Tableau 3 : propriétés de ConectionString

### Le cryptage :

Le cryptage est effectué par un algorithme de cryptage symétrique c'est-à-dire un algorithme de cryptage décryptable. Ce cryptage doit être décryptable parce que la connexion à la base de données doit savoir relire le mot de passe.

Plusieurs algorithmes symétriques sont disponibles avec le *dotNET Framework*.

- DES
- RC2
- Rijndael
- TripleDES

Ces algorithmes symétriques (ou à clé secrète) utilisent une clé de cryptage et un vecteur d'initialisation pour protéger les données.

Pour réussir à décrypter les données avec ce genre de cryptage, il faut absolument enregistrer la clé et le vecteur d'initialisation. Si on perd la clé de cryptage, on ne pourra jamais plus décrypter les données.

Le vecteur d'initialisation ne sert qu'à empêcher que deux chaînes de caractères identiques ne soient cryptées de la même manière.

### **Le formulaire adresse**

The image shows a screenshot of a web application window. The window title bar is blue and contains the text 'Adresse' on the left and standard window control icons (minimize, maximize, close) on the right. The main content area has a light gray background. At the top center, the text 'Entrer L'adresse du site' is displayed in a serif font. Below this, the text 'Tapez l'adresse du site :' is followed by a white text input field. In the bottom right corner of the form area, there is a rectangular button with the text 'Envoi'.

**Fig 9 : Formulaire adresse**

Ce formulaire permet à l'utilisateur de rentrer ou de modifier l'adresse URL du site d'administration du serveur syslog.

Ce formulaire apparaît lorsque l'utilisateur clique sur le menu "*view administration*", pour lui demander de rentrer l'adresse du site d'administration. Une fois cette adresse validée, le site s'ouvre dans Internet explorer et l'adresse est enregistrée dans un fichier texte.

Si l'accès au page d'administration est modifié pour une raison ou pour une autre, l'utilisateur peut aller le préciser en cliquant sur le menu "*adresse du site*", ce qui échangera l'adresse qui se trouve dans le fichier.

## La base de données

La base de données est réalisée grâce à SQL Server 2000. Cette base de données est composée de trois tables :

- "\_syslogtable" : table contenant les messages.
- "\_codeseverity" : table contenant les code de *severity*
- "\_codefacility" : table contenant les codes de *facility*.

## Difficultés rencontrées

Les difficultés rencontrées lors de la création de ce programme ont été

- L'apprentissage d'un nouveau langage de programmation car les langages utilisés par le CEPS sont deux langages que nous n'avions pas appris durant nos trois ans d'études.
- La réunification de la documentation sur les serveurs syslog et compréhension sur le fonctionnement de ces serveurs.
- La recherche d'informations concernant la meilleure méthode pour programmer en Visual C++.
- Remarquer que les fabricants ne respectent pas les normes censées leur permettre de mieux faire communiquer leurs matériels entre eux.
- Le débbugage du programme une fois qu'il a été mit en production sur le serveur de l'entreprise car le fait que le programme fonctionne en local sur notre machine ne veut pas forcément dire que cela va fonctionner sur une autre machine.

## 5.

## **Le site d'administration**

Le site d'administration est une interface Web développée en ColdFusion qui permet à l'administrateur réseau de visualiser les différents messages qui transitent dans son réseau. Il est bien plus agréable de visualiser les messages, via une interface graphique qui s'en va les rechercher dans la base de données plutôt que de devoir aller les regarder en brut dans le gestionnaire de données. De plus l'interface web nous permet aussi de rajouter des fonctions pour agir sur les messages sans que l'utilisateur ne se rende compte de quoi que ce soit.

Il est composé de plusieurs parties pour agir avec les messages stockés dans la base de données, et pour naviguer dans l'interface.

- La page d'accueil
- Tout voir
- Recherche
- Purge
- Statistiques

### **La page d'accueil**

Sur la page d'accueil, un message d'accueil est présent pour informer l'utilisateur du but du site. En plus de ce message, en guise de mode d'emploi, l'utilisateur peut y voir un plan du site.

### **Tout voir**

Sur cette page, une requête de sélection nous permet de visualiser tous les messages de la base de données. Ces messages sont affichés à l'aide d'un tableau.

La requête est "*SELECT \* FROM \_syslogtable*", elle permet de sélectionner tous les messages présent dans la table *\_syslogtable*.

### **La Recherche**

La recherche se fait à l'aide de différents filtres :

- Par severity
- Par facility
- Par Hostname
- Par date
- Recherche avancée

### Par severity

Dans la page de recherche, il y a un formulaire qui contient une liste déroulante et un bouton. La liste déroulante est remplie à l'aide d'une requête, par les phrases correspondantes à la *severity*. Ces phrases sont contenues dans une des tables de la base de données.

L'utilisateur sélectionne donc le code de *severity* dans la liste déroulante et ensuite clique sur le bouton Search. L'action du formulaire est de renvoyer l'utilisateur sur une autre page sur laquelle les messages, correspondant à la *severity* sélectionnée, sont affichés dans un tableau.

### Par facility

Dans la page de recherche, il y a un formulaire qui contient une liste déroulante et un bouton. La liste déroulante est remplie à l'aide d'une requête, par les phrases correspondantes à la *facility*. Ces phrases sont contenues dans une des tables de la base de données.

L'utilisateur sélectionne donc le code de *facility* dans la liste déroulante et ensuite clique sur le bouton Search. L'action du formulaire est de renvoyer l'utilisateur sur une autre page sur laquelle les messages, correspondant au code sélectionné, sont affichés dans un tableau.

### Par hostname

Comme pour les pages précédentes, cette page contient donc un formulaire avec une liste déroulante et un bouton. Cette fois-ci, la liste déroulante est remplie à l'aide du champ *hostname* de la base de données. Une fois que l'utilisateur clique sur le bouton Search, le formulaire le renvoie sur une page qui affiche tous les messages qui ont été envoyé par l'appareil correspondant à l'adresse sélectionnée.

### Par date

Cette page est composée de deux formulaires, le premier pour effectuer une recherche des messages à une date bien précise et le second pour effectuer la recherche des messages entre deux dates. Les formulaires renvoient tous les deux sur une page d'affichage qui permet de visualiser les messages dans un tableau.

### Recherche avancée.

La recherche avancée contient deux formulaires, un avec des listes déroulantes, l'autre avec une zone de texte. Dans le premier formulaire, on peut choisir la *severity*, la *facility* et la date du ou des messages que l'on veut voir.

Dans le second, il suffit à l'utilisateur de taper un mot clé ou une phrase et la recherche s'effectuera sur tous les champs de la table.

L'affichage des messages se fait comme pour les autres pages dans un tableau.

## La purge

La purge se fait grâce aux mêmes filtres que pour la recherche c'est-à-dire

- par facility,
- par severity,
- par hostname,
- par date,
- par selection
- tout effacer.

La seule différence avec la recherche c'est que les formulaires ne renvoient pas sur un tableau affichant les messages mais sur une page avec un message spécifiant que les éléments ont bien été effacés.

## Les statistiques

La page de statistiques, contient des graphiques qui représentent le calcul du nombre de messages présent dans la base de données. Pour chaque *severity*, il y a un calcul du nombre de message ayant la même *facility*. Une fois ce calcul effectué, les données sont retransmises à l'utilisateur sous forme de graphique.

## Les difficultés rencontrées

Les difficultés rencontrées pour le site ont été

- La réalisation du design du site : Le design est réalisé en CSS car si on veut respecter les standards du web quelques éléments de l'html doivent être proscrit le plus possible sauf dans des cas de grandes nécessités. Pour la mise en page, les tableaux sont des éléments de l'html qui ont quasiment disparu, les "frames" sont totalement à proscrire. De ce fait, les feuilles de style en cascade (CSS) sont une alternative qui commence à voir le jour dans le développement web. Il a donc fallu apprendre comment réaliser un site à l'aide de ces feuilles de style.
- Le respect des standards du web. Et surtout, rendre le site accessible aux nouveaux navigateurs et pas seulement à Internet Explorer. De nos jours, même si Internet Explorer est le navigateur le plus répandu, il tend à disparaître et à laisser sa place aux navigateurs "libres" (Mozilla Firefox). Aux vues de cette augmentation, il faut arriver à réaliser un site qui peut être visionné aussi bien par Internet Explorer que par les autres.

6.

## Sql Server 2000

Microsoft SQL Server est une solution complète de base de données et d'analyse conçue pour le développement d'applications, ....

### Ses bénéfices

- Entièrement conçu pour le WEB : Effectuer des requêtes et des analyses de données en toute simplicité sur le WEB. Accéder facilement et en toute sécurité aux données à partir d'un navigateur.
- Evolutif et fiable : Répartir la charge de votre base de données pour obtenir une montée en puissance linéaire de votre application.
- Rapidité de mise en œuvre.

### Ses types de données

- Entiers :
  - Bigint : Entier signés de 64 bits.
  - Int : Entier signé de 32 bits.
  - Smallint : Entier signé de 16 bits.
  - Tinyint : Entier non signé de 8bits.
- Binaire :
  - Bit à 0 ou 1
- Décimaux et numérique :
  - Decimal : nombre à précision fixe
  - Numeric : équivalent au type décimal
- Monétaires :
  - Money : représente une somme (4 chiffres après la virgule)
  - Smallmoney : équivalent à money mais avec un plus petite precision.
- Numérique approchés :
  - Float : nombre à virgule flottante
  - Real : nombre à virgule flottante de grande précision.
- Date :
  - datetime : date et heure allant du 1<sup>er</sup> Janvier 1753 au 31 Décembre 9999 avec une précision de 3,33milliseconde.
  - Smalldatetime : Date allant du 1<sup>er</sup> Janvier 1900 au 6 Juin 2079 avec une précision d'1 minute.
- Chaîne de caractères :
  - char : chaîne de caractères fixes (8000 caractères max)
  - varchar : chaîne de caractères à longueur variable (8000 caractères max)
  - text : chaîne de caractères de taille pouvant aller jusqu'à  $2^{31}-1$  caractères.

- Chaîne de caractères incluant les caractères UNICODE :
  - nchar : chaîne de caractères fixe (8000 caractères max)
  - nvarchar : chaîne de caractères de longueur variable (8000 caractères max)
  - ntext : chaîne de caractères de taille pouvant aller jusqu'à  $2^{31}-1$  caractères.
- Chaîne binaire :
  - Binary : chaîne binaire de taille fixe (8000 octets max)
  - Varbinary : chaîne binaire de taille variable (8000 octets max)
  - Image : chaîne binaire d'une taille pouvant aller jusqu'à  $2^{31}-1$  octets.
- Autres :
  - Cursor : référence vers un curseur.
  - Sql\_variant : pouvant contenir tous les types sauf : text, ntext, image et sql\_variant.
  - Table : type utilisé pour stocker des résultats pour une utilisation ultérieure.
  - Timestamp : nombre unique mis à jour à chaque jour d'un enregistrement.
  - Uniqueidentifier : identifiant global unique de type GUID.

## **7. Conclusion**

Avant ce stage, je ne connaissais que ce que j'avais eu la chance d'apprendre à l'école, je ne savais pas ce qu'était réellement l'informatique professionnelle. Je ne savais pas du tout comment pouvait se dérouler la réalisation d'une application professionnelle.

Grâce à ces quatre mois dans une entreprise, j'ai eu la possibilité d'apprendre beaucoup de choses en informatique, d'approfondir mes connaissances personnelles. J'ai aussi pu réaliser que ce domaine demande énormément de temps pour parfaire son apprentissage et que la constante évolution des technologies réclame qu'un informaticien se tienne au courant assez régulièrement des nouveautés.

Ce stage m'aura donc permis d'accroître les connaissances acquises lors de mes années d'études. Et de m'intéresser un peu plus à des langages de programmation différents, que ce soit en programmation logicielles ou en développement web. Mais, il m'aura aussi et surtout permis d'acquérir une expérience professionnelle non négligeable pour mon futur.

## **Bibliographie :**

### ▪ **Ouvrages Consultés**

- ALLAIRE, Developing Web Applications with ColdFusion, Allaire Corporation
- CHAPMAN D., *Microsoft Visual C++6*, CampusPress.(Collection Le programmeur)
- LIBERTY J., Le langage C++, CampusPress (Collection Le programmeur)
- MSDN Library pour Visual Studio.NET 2003

### ▪ **Sites Consultés**

- Internet RFC/STD/FYI/BCP Archives (site consulté le 21/02/2005).  
Adresse URL : <http://www.faqs.org/rfcs/>.
- Microsoft visual C++ Developer Center (Site consulté à plusieurs reprises)  
Adresse URL : <http://msdn.microsoft.com/visualc/>.
- C/C++ Codes sources (Site consultés à plusieurs reprises)  
Adresse URL : <http://cppfrance.com/> .
- Developpez.com Le club des développeurs (Site consultés à plusieurs reprises)  
Adresse URL : <http://www.developpez.com/> .
- Open Web Group pour les standards du Web (Site consultés le 07/05/2005)  
Adresse URL : <http://openweb.org.eu/> .
- Alsacréations (Site consultés à plusieurs reprises)  
Adresse URL : <http://css.alsacreations.com/> .