**Introduction**

Les différents équipements qui composent un réseau informatique (routeur, switch, firewall) sont amenés à générer différents messages systèmes. Ces messages générés par le système d’exploitation de ces appareils peuvent être des messages d’information ou bien des messages d’erreur nécessitant une intervention.

Plusieurs solutions existent pour consulter ces messages :

* Consulter directement la console de l’équipement en question
* Consulter à distance la console via SSH ou Telnet
* Envoyer les messages à un serveur Syslog
* Convertir les messages en format SNMP et les envoyer à un Network Management Server

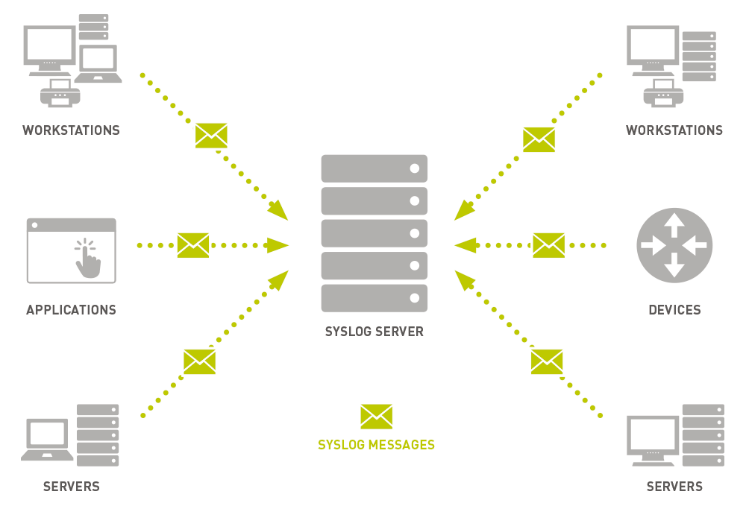
Le choix a été fait d’utiliser le modèle Syslog pour monitorer ces équipements. L’intérêt de Syslog est de centraliser les journaux d’évènements des différents équipements afin d’avoir une vision globale du réseau et d’être informé en temps réel de son état.

**Syslog**

Syslog est le système de journal d’événement sous UNIX. Il permet de transporter sur le réseau les messages des différents équipements vers un serveur Syslog. Il passe par le port 514 ( UDP)

Ce protocole définit 3 éléments :

* Un périphérique (routeur, switch, firewall …) ou une application qui envoi des messages Syslog
* Un relais optionnel qui reçoit les messages et les envois vers un autre équipement
* Un collecteur qui reçoit les messages uniquement (ex : serveur Syslog)



Il définit également 3 notions liées à la priorité d’un message :

- facility : concerne les applications et correspond à leur type parmi 24 possibles

- severity : correspond au niveau d’importance du message parmi 8 possibles

- priority : <val> où val = (8\* facility) + severity

|  |  |
| --- | --- |
|  |  |

Trame Syslog

Une trame Syslog se présente sous la forme de 3 champs :

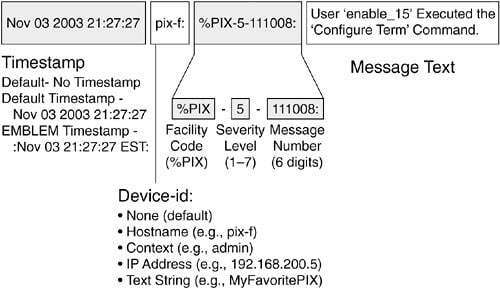
|  |  |  |
| --- | --- | --- |
| PRI | HEADER | MSG |
| facility | date et Hostname | message d’information qui doit être envoyé |

Exemple  de trame:

<165>May 18 14:46:18 192.168.1.1 Un message Syslog classique

|  |  |  |
| --- | --- | --- |
| PRI | HEADER | MSG |
| <165> | May 18 14:46:18 192.168.1.1 | Un message Syslog classique |

RFC3164 INITIALE - RFC 5424 actuelle : compatibles entre elles



Syslog appliqué aux équipements réseaux

Les message syslogs ne sont pas uniformisés ; ils changent en fonction de l’équipement, du fabriquant… Ici l’exemple d’un message CISCO IOS qui se présente dans le format suivant :

**timestamp**%<**facility**>-<**severity**>-<**mnemonic**>: <**message-text**>

Example:

|  |
| --- |
| \*Jan 18 03:02:42: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/0, changed state to down |

Il se décompose en 3 parties :

|  |  |  |
| --- | --- | --- |
| Date de l’événement | TAG Syslog | Description sommaire |
| Jan 18 03:02:42 | %LINEPROTO-5-UPDOWN | Line protocol on Interface GigabitEthernet0/0, changed state to down |

Le TAG Syslog donne également des informations supplémentaires :

|  |  |  |
| --- | --- | --- |
| Source émettrice du message (routeur …) | Criticité (0-7) | Mnémonique unique pour ce message |
| %LINEPROTO | 5 | UPDOWN |

Les niveaux de criticité sont les mêmes que pour la trame Syslog initiale, sans facility étant donné que ce ne sont pas des applications.

Il est possible de configurer les équipements réseaux pour qu’ils envoient des Syslog avec une severity précise. Par exemple les syslogs d’informations ne sont pas forcément utiles à faire remonter au serveur.

Problématiques de Syslog en production

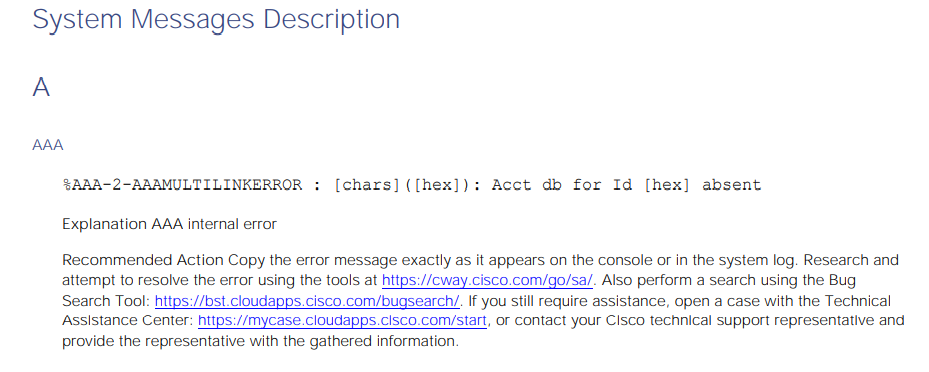
Le 1er niveau de diagnostic est très compliqué pour identifier les évènements (Syslog, Alertes de Monitoring) ce qui rallongent le délai de résolution des incidents. Pour identifier rapidement un Syslog il faut, à partir du TAG associer différentes informations :

* La description minimale du Syslog en question **< Description >**
* Une description plus détaillée du message **<Explanation >**
* Des recommandations concernant les potentielles causes et résolutions possible pour cet incident **< Recommendations >**
* Un niveau de sévérité permet d’identifier rapidement les messages importants dans la masse de Syslog reçue en permanence. Ce niveau peut être différent de celui initialement contenu dans le syslog **<Severity>**

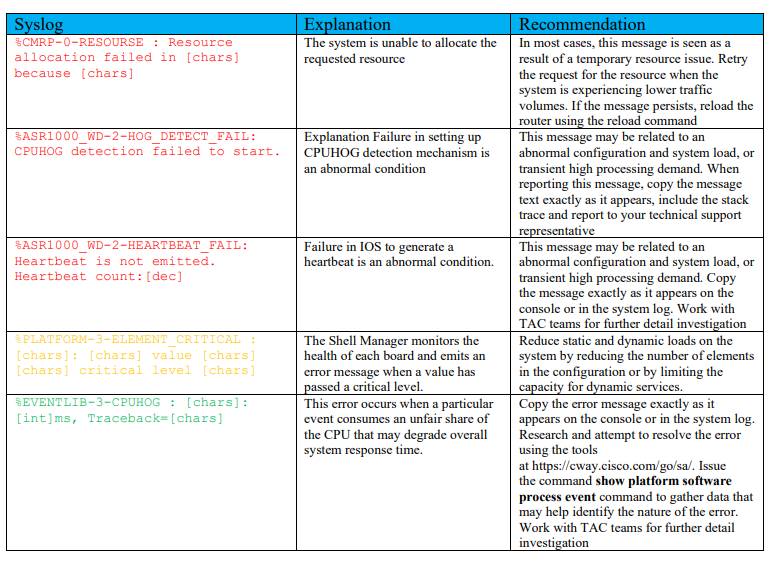
Exemple :

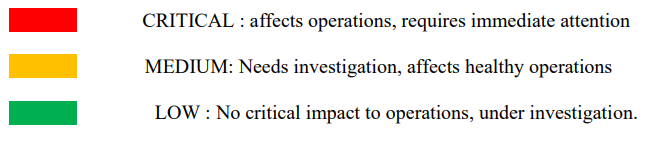
|  |  |  |  |
| --- | --- | --- | --- |
| **< Description >** | **<Explanation >** | **< Recommendations >** | **<Severity>** |
| Resource allocation failed in [chars] because [chars] | The system is unable to allocate the requested resource | In most cases, this message is seen as a result of a temporary resource issue. Retry the request for the resource when the system is experiencing lower traffic volumes. If the message persists, reload the router using the reload command | Critical |

Actuellement ces informations sont généralement contenues dans des PDF envoyés par les fabricants :



Certains ajoutent en plus une criticité supplémentaire qui ne correspond pas à celle initialement dans le format Syslog :





Cela implique une recherche fastidieuse dans ces documents souvent incomplets et qui existent sous plusieurs versions ce qui ralenti le travail de gestion des incidents.

L’objectif pour palier à ce problème est donc de créer une application web pouvant stocker l’ensemble de ces données soit sous la forme d’une table unique s’il est possible d’uniformiser les Syslog, soit par type d’équipement réseau (une table Cisco IOS, une table Cisco ASA …).

Ainsi il sera possible :

* De consulter les informations liées à un Tag Syslog
* D’avoir l’accès à l’intégralité des Syslog pour un équipement donné
* De trier les données recherchées par niveau de sévérité
* De récupérer le contenu d’une table au format CSV ou JSON
* De modifier des entrées pour compléter/mettre à jour des informations manquantes
* D’accéder à la base via une API sans passer par l’interface graphique

Cette base pourra donc évoluer pour contenir des informations Syslog sur de nouveaux équipements et de corriger des données existantes.

Une évolution possible serait de pouvoir directement lancer un ensemble de commandes vers les équipements réseaux pour collecter des données complémentaires.

Un autre point important à étudier dans la création de cette base est l’intégration des données. Certains documents contiennent plus de 2000 lignes, il faut donc une solution pour les intégrer directement dans la base plutôt que de devoir les formater une par une (parser).

Réalisations du projet

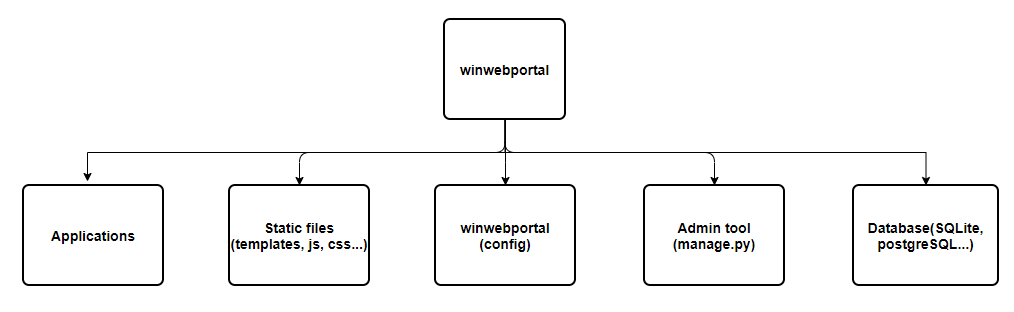
1/ Principe de fonctionnement

Pour réaliser ce projet dans le laps de temps de ce stage, la solution de Django était la plus adaptée.

Django est un Framework open source qui permet de développer rapidement un site web avec une base de données intégrée. Il inclut de nombreuses fonctionnalités et est très polyvalent. Le fait qu’il s’agissent d’un Framework très populaire permet également de bénéficier de beaucoup de documentation.

Ainsi lors de la première partie du stage j’ai pu trouver rapidement de nombreux tutoriels et cours en ligne pour me former sur ce Framework. Cela m’a permis après quelques petits projets d’introduction à Django de commencer directement le développement pour le projet.

Un projet Django s’organise dans un répertoire de cette forme :

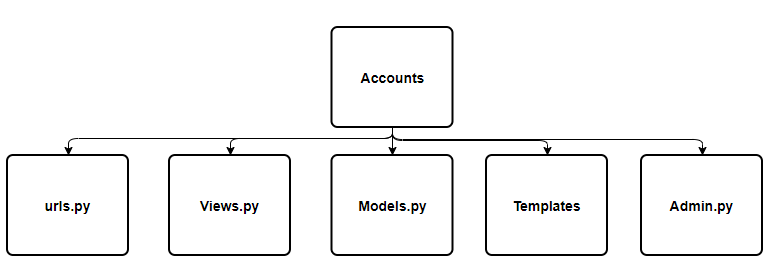


Le répertoire principal “Winwebportal” contient :

* Un fichier manage.py qui est utilitaire admin permettant de démarrer le projet, ajouter des applications, ajouter des superusers, faire la migration des tables dans la base de données …
* Un fichier/répertoire de base de données du projet. Par défaut SQLite est utilisée mais on peut le remplacé par n’importe quelle autre.
* Un répertoire de fichiers statiques pour toutes les applications tierces utilisées. Il contient les fichiers CSS, JS, HTML fournis par ces modules externes qui seront utiles pour nos applications
* Un second répertoire winwebportal qui contient un fichier de configuration settings.py qui indique la base de données utilisée, la liste d’applications installées … et un premier fichier de redirection des URLS urls.py

Ce projet Django s’organise autour « d’applications » qui ont chacune leur répertoire. Dans notre cas une application « table » pour les Syslog et une application « accounts » pour l’authentification ont été créés.

Le répertoire classique d’une application se présente sous cette forme :



Il contient :

* Un fichier models.py qui décrit le schéma souhaité pour les tables que l’ont veut avoir dans la base de données
* Un fichier views.py qui va contenir toutes les méthodes de l’application. Elles ont pour rôle d’effectuer le traitement des données (récupérer l’intégralité de la base, supprimer un Syslog …) et d’indiquer quelle page HTML utiliser pour l’affichage…
* Un fichier urls.py qui est le fichier qui va relier l’URL demandée à la méthode correspondante dans views.py
* Un dossier « Templates » qui va contenir toutes les pages HTML de l’application pour l’affichage
* Un fichier admin.py qui permet d’ajouter les tables de l’application à la page admin de Django

Django a une fonctionnalité très intéressante : il possède un page d’administration préexistante qui permet de gérer ses modèles, les permissions des utilisateurs… Cela apporte un gain de temps considérable sur le développement. Ici un exemple sur une page d’administration créée automatiquement pour un projet de bibliothèque :

Django fonctionne sur une variante du modèle d’architecture MVC, le modèle MVT ( Modele, Template, View).



Le principe de fonctionnement complet est le suivant :

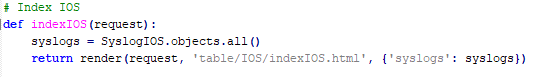
1. L’utilisateur demande à accéder une adresse URL depuis son navigateur (ex : tous les syslogs d’IOS)



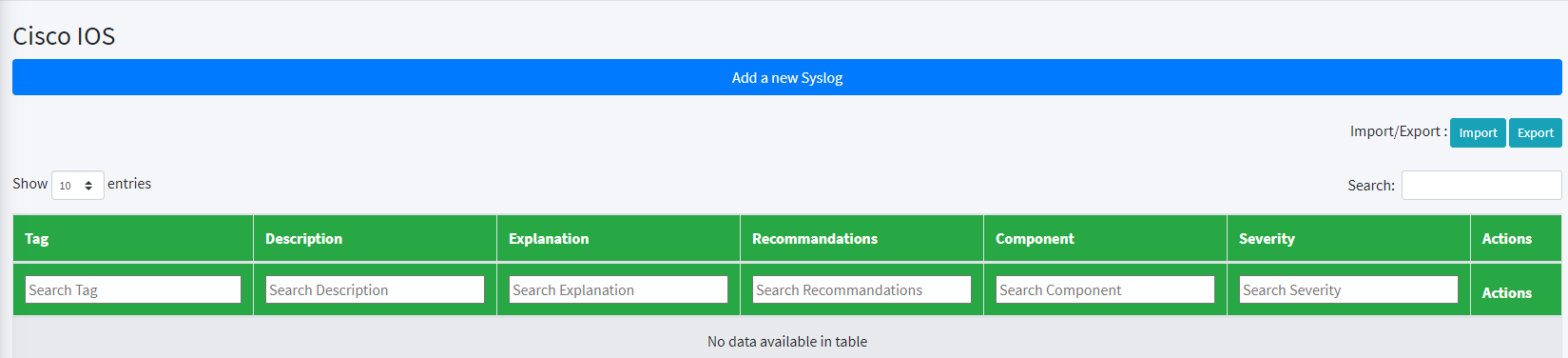
1. Au niveau de django, le fichier urls.py redirige la requête vers la méthode correspondante dans views.py (ici indexIOS)



1. La méthode appelée effectue les différents traitements sur les données dans la base avec l’ORM django qui fait le lien entre la base et l’application (il récupère tous les Syslogs dans la table SyslogIOS définie dans models.py) puis il appelle la page correspondante.



1. La méthode effectue les différents traitements à réaliser



2/ Réalisation de la solution

Django correspondait donc parfaitement pour la réalisation de ce projet avec la base de données qui contiendrait tous les syslogs et la partie web qui servirait d’interface pour les personnes ayant besoin de la consulter.

Initialement il y avait trop peu de données pour définir un schéma unique pour les tables de Syslogs, les 2 seuls exemples étant Cisco ASA et Cisco IOS pour le routeur ASR1K. Les 2 schémas choisits étaient donc :

|  |
| --- |
| Table IOS |
| TAG  Description  Explanation  Recommendations  Model  SeverityIOS |

|  |
| --- |
| Table ASA |
| TAG  Description  Explanation  FEATURE  Recommendations  Severity |

A partir de ces définitions les données fournis par Cisco nécessitait un traitement pour corriger toutes les erreurs, supprimer les champs redondants et tout transformer dans le format UTF-8. Une partie a été faite à l’aide d’un parseur en python, le reste nécessitait un travail manuel.

Par la suite, ayant reçu d’autres exemple de Syslog de la part de Cisco j’ai pu définir un schéma unique pour ces syslogs :

|  |
| --- |
| Syslog model |
| TAG  Description  Explanation  Recommendations  Component  Severity |

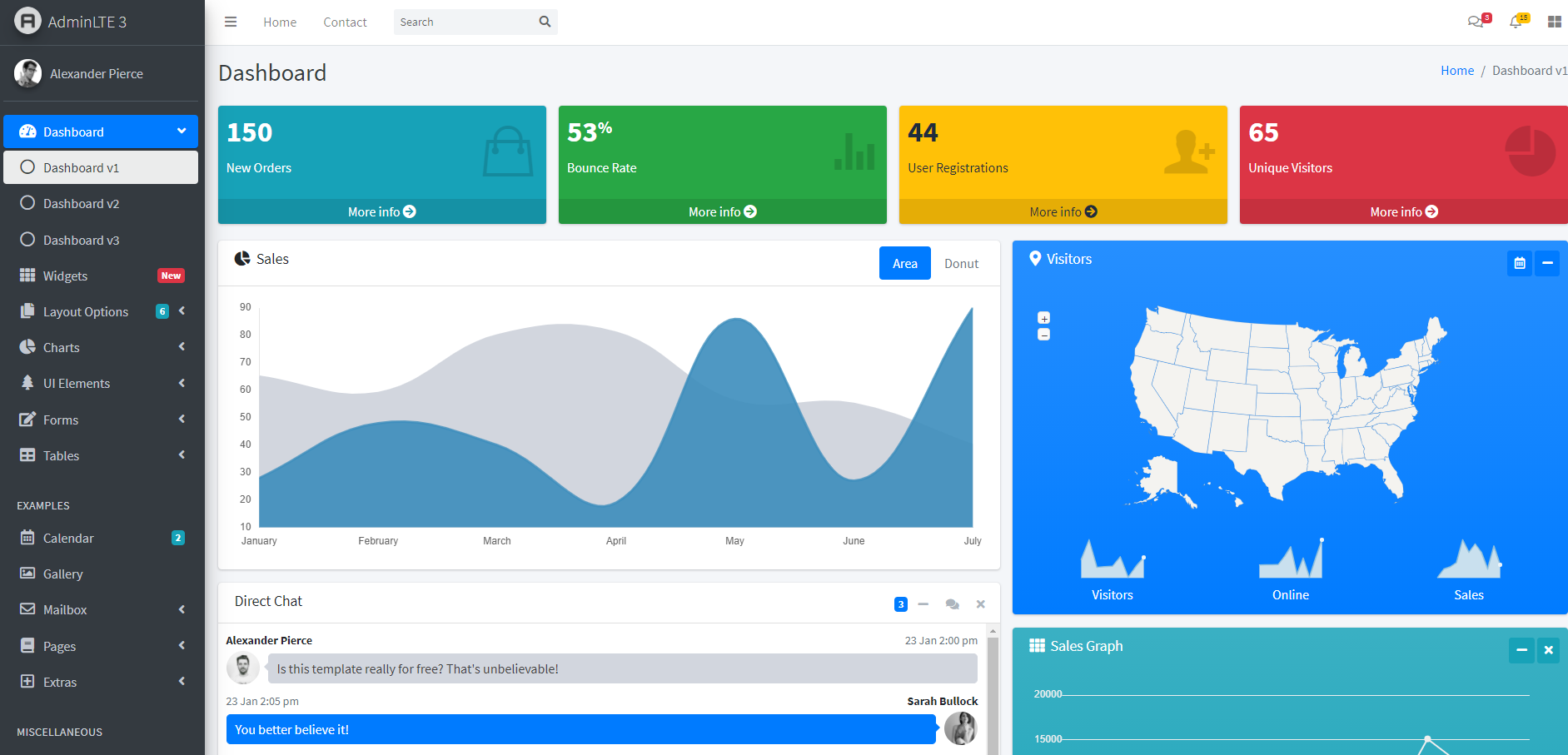
De plus, la criticité différente initialement présente sur Cisco IOS a été supprimée après discussion avec le fournisseur. Les 8 niveaux de criticité normaux pour Syslog ont été utilisé à la place.

Une fois les données mises en forme, il fallait définir l’interface graphique à utiliser.

Interface utilisateur :

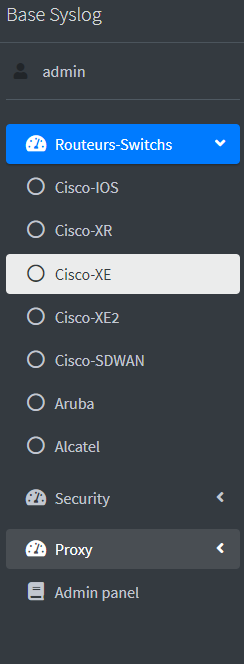
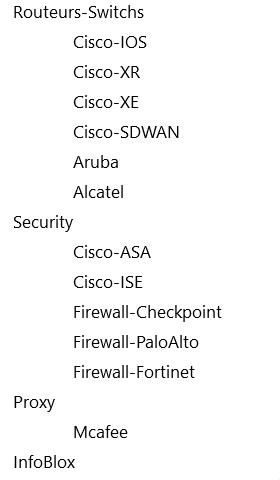
Pour l’interface principale j’ai utilisé AdminLTE-3 qui est un Template orienté interface administrateur/ tableau de bord.

Exemple de page réalisée avec AdminLTE-3



C’est un template très complet en termes de documentation et il propose de nombreux Template pour voir les différentes pages que l’on peut réaliser avec.

C’est la base qui a servi à la réalisation de l’interface pour les menus. Cisco a fourni une liste de tous les syslogs pouvant être intégrés à la base ce qui m’a permis de créer une hiérarchie complète des menus même en ayant seulement 4 ressources Syslog au départ.

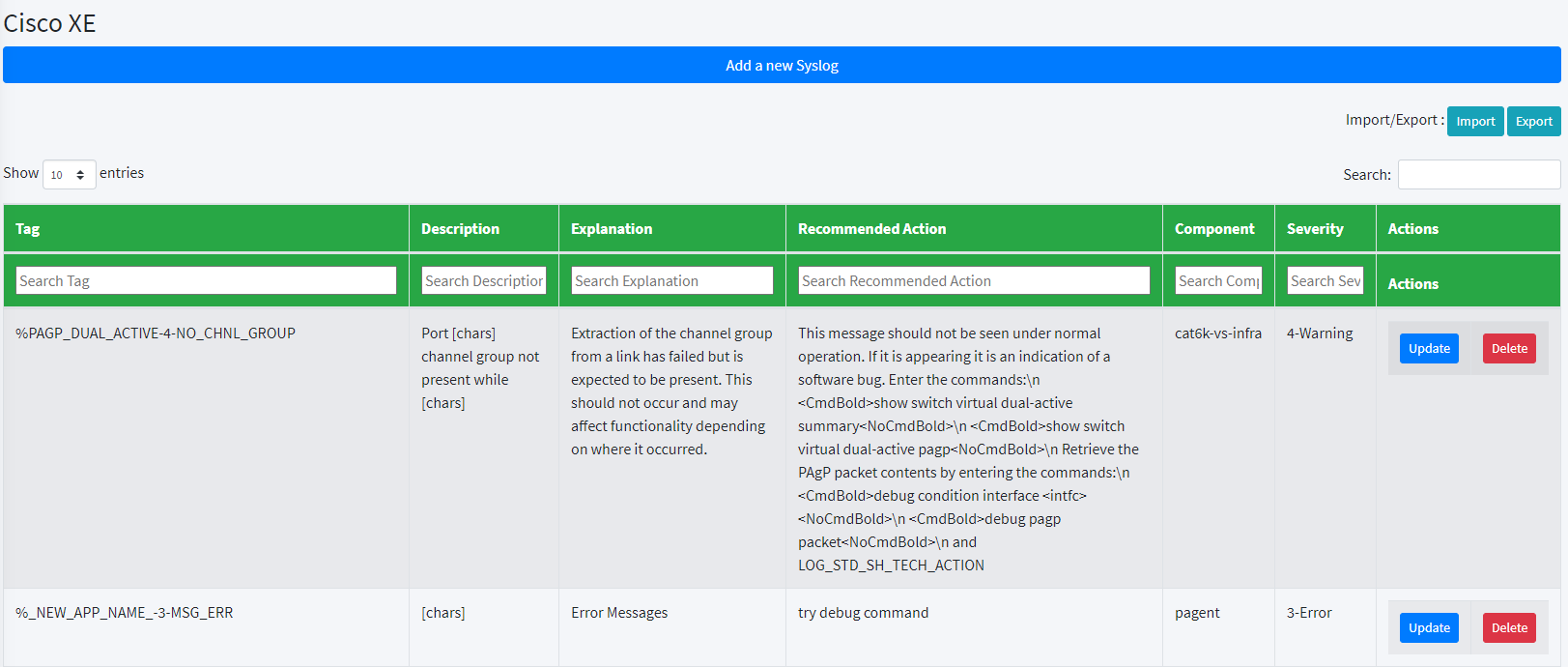


Mise en page des données

Initialement l’affichage des données était réalisé par une simple table HTML avec un champs de recherche dédié. Cet affichage convient pour quelques lignes de données mais n’est pas adapté aux milliers de lignes de syslogs. Il éta

Datatables qui est un plug-in permettant de rajouter beaucoup de fonctionnalités rendant les tables très interactives. Ainsi elle m’a permis de créer des champs de recherche individuels pour chaque colonne tout en gardant un champ de recherche général. Les données sont également décomposées sur plusieurs pages avec le choix de décider combien sont affichées.

Voici l’interface obtenue :



CRUD

Les champs sont toujours dans le même ordre quel que soit le fabriquant. A chaque fin de ligne sont regroupées les fonctions de suppression et d’édition

La suppression affiche un message de confirmation avec le nom du Syslog afin d’éviter toute mauvaise manipulation

La fonction d’édition permet de changer tous les champs excepté le TAG étant donné qu’il s’agit de la clef primaire (un nouveau Syslog serait créé au lieu de modifier l’existant)

|  |  |
| --- | --- |
| Edition | Création |
|  |  |
| Suppression | |
|  | |

Import / Export

Une autre fonctionnalité demandée, en plus de la simple création de Syslog, était la possibilité d’importer/exporter des données via des CSV. En effet la plupart du temps les bases sont créées une fois et peuvent contenir plusieurs milliers de lignes ce qui est fastidieux à ajouter.

Pour cela j’ai utilisé la librairie Django-import-export qui permet d’effectuer ce genre de tâches dans plusieurs formats (Excel, CSV, JSON). Elle permet également d’avoir une « preview » des données qui vont être ajoutée et supporte également la mise à jour des données. Ainsi si le CSV d’un certain type de Syslog est mis à jour, on peut directement injecter le nouveau plus complet.

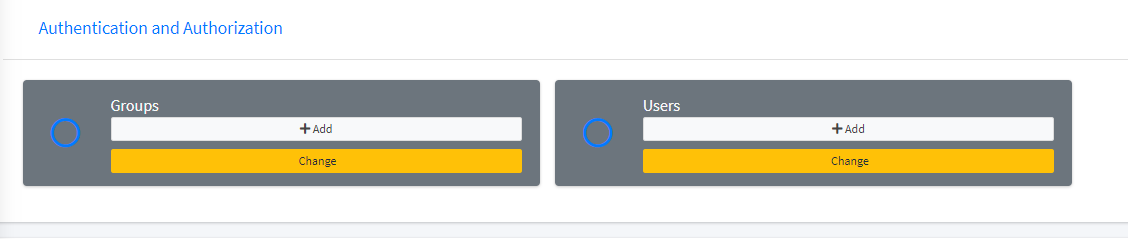
|  |
| --- |
| Import |
|  |
| Preview |
|  |
| Export |
|  |

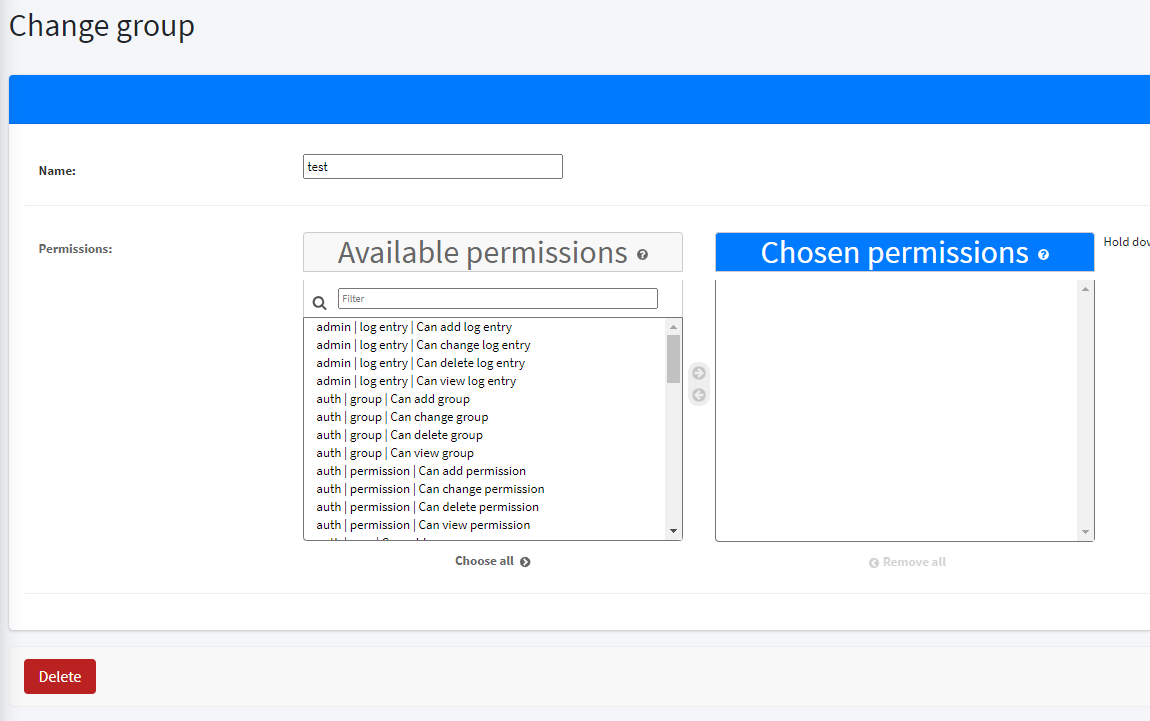
Utilisateurs et autorisations

Un système d’authentification a également été créer afin de hiérarchiser les accès à l’application :

* Un utilisateur non loggé peut uniquement consulter la base
* Un utilisateur sans droit peut consulter et exporter la base
* Un utilisateur avec les droits peut créer, modifier, importer et exporter
* Un admin a tous les pouvoirs et peut accéder à la page d’administration

Les droits des utilisateurs sont créés avec la page « groups » de Django admin qui permet de définir leurs permissions une fois qu’ils sont ajoutés dans un groupe :





Affichage correspondant à chaque type de connexion :

|  |
| --- |
| Unlogged |
|  |
| Logged |
|  |
| Editor |
|  |

Améliorations :

À la suite de ce premier projet Django, des améliorations sont possibles :

Les données de Syslog XE et XR contiennent chacune 15 000 lignes ce qui alourdit significativement le chargement de la page (environ 1 minute). Une solution face à des tables aussi lourdes est de passer Datatable côté serveur (auparavant le traitement des données se faisait ). Ainsi on peut limiter le nombre de données envoyées (ex : 500) tout en gardant les fonctionnalités de recherche

Creation d’une API