

# Table des matières

[**Table des matières**](#_jz9uxxii8lgc) **1**

[**Analyse préliminaire**](#_gjdgxs) **2**

[Introduction](#_30j0zll) 2

[Organisation](#_1fob9te) 2

[**Analyse**](#_3znysh7) **2**

[Analyse du cahier des charges](#_ln9ti47ctlvu) 2

[Analyse concurrentielle](#_mkg61zafd484) 2

[Cahier des charges détaillé](#_iy93mqaawuwn) 4

[Travail à réaliser](#_mwo43f6s5cdf) 4

[Livrables](#_kjzqupq99svb) 4

[Maquettes préliminaires](#_ntk74qr18ydo) 4

[Planification](#_y66mijpjov8y) 6

[**Conception**](#_1ci93xb) **7**

[Analyse de l’environnement](#_fed5wd56zg7k) 7

[Les logiciels utilisés](#_982esbtj8d0g) 7

[Matériels utilisés](#_6267w7nhz9sj) 7

[Langages utilisés](#_iulsjfduu5lz) 7

[Sauvegardes et backups](#_uuxya8nxfaa1) 7

[**Conception du code**](#_nxesldrz7ikc) **8**

[Java](#_nribkknbbq6h) 8

[Python](#_z54dxmtnsebe) 9

[Bash](#_n2jb1whppr1d) 9

[**Ressources**](#_e223b7yoqt9e) **9**

[Librairies utilisées](#_40sz3xrez8zt) 9

[**Conclusion**](#_d04o1qaexv4o) **10**

[Réalisation du projet](#_ma2ibp7t3u3n) 10

[Résultat obtenu](#_4j0tq5f733e1) 10

[Améliorations possibles](#_23qj4payl0nz) 10

[**Annexes**](#_2yrz4dehiv9e) **11**

# Analyse préliminaire

## Introduction

Dans le cadre de la formation Technicien ES au CFPT-Informatique, il nous a été demandé de réaliser un projet nommé *NSAei*. Nous devons fournir comme livrable, au plus tard le lundi 19 mars à 11h30, une documentation techniques ainsi qu’un manuel d’utilisation. Il y aura également une présentation suivie d’une démonstration.

## Organisation

**Étudiants : Client :**

Schito Dylan C. Maréchal  
 Troller Fabian Chauche Benoit

Les présentations se dérouleront le lundi 19 mars à partir de 12h30 en salle R104 dans le bâtiment Rhône. Elles seront scindées en trois partie, 10 minutes de présentation, 5 minutes de démonstration et 10 minutes de questions des personnes présentes.

# Analyse

## Analyse du cahier des charges

Le but de ce projet est de pouvoir utiliser un Raspberry Pi 2 comme analyseur de réseau en se connectant à une prise Ethernet. Il sera possible de visualiser toutes les informations relatives à celle-ci sur l’interface utilisateur de l’application.

## Analyse concurrentielle

Notre cahier des charges est basé sur un produit déjà existant nommé NSA (Network Statut Analyzer).

Ce produit permet les fonctionnalitées suivantes, tirées du site du constructeur[[1]](#footnote-0) :

* **Analyse**   
  Analyse et documente les réseaux complets. Le résultat peut être consulté dans les navigateurs Web ou téléchargé en format PDF / DVI à partir du lecteur.
* **TFTP Server**Le "Network Status Analyzer" dispose d'un serveur TFTP, vous pouvez donc sans outils, commutateurs, routeurs, téléphones et autres matériels avec un nouveau firmware ou une configuration de magasin
* **Serveur DHCP**   
  "Network Status Analyzer" est également un serveur DHCP si vous en avez besoin. Par exemple, pour fournir une configuration ou un micrologiciel au démarrage.
* **Statistiques**   
  Cela vous permet de mesurer le débit de la bande passante de chaque carte réseau ou chaque port de commutation individuels et diagnostiquer ainsi les goulots d'étranglement du réseau capable par tout dispositif de réseau SNMP (commutateur, pare-feu, routeur, ...).
* **Packet Generator**   
  Génère des paquets de données qui peuvent ensuite être suivis avec un renifleur. Cela vous permet de tester les listes de contrôle d'accès et de localiser très rapidement les erreurs dans le réseau (où le paquet généré arrive-t-il encore, où pas).
* **WebGUI sniffer**  
  Qui renifle directement sur l'affichage Network Status Analyzer ou via l'interface graphique Web.
* **Sondage WLAN**   
  Vous pouvez parcourir une zone (bâtiment) à l'aide de l'analyseur d'état du réseau, puis stocker et analyser (plus tard) les qualités actuelles du signal WLAN aux points saillants.
* **Router Discover**   
  Vous pouvez trouver tous les routeurs et la passerelle par défaut. Ceci est particulièrement intéressant si vous voulez démasquer des attaquants dans le réseau local.
* **WinShare Discover**   
  Qui offre quel lecteur Windows?
* **Commande et contrôle à distance**   
  Ainsi, l'être contrôlés « Analyzer état du réseau » à distance via SSH sur Internet sans un tunnel VPN ou SSH dans le réseau dans lequel il est situé à besoin. Cela nécessite un abonnement au service cloud à schaffer-se.

## Cahier des charges détaillé

### Travail à réaliser

Les fonctions (non-exhaustive) à mettre en place sont :

* Analyse
* TFTP Server
* DHCP Server
* Statistik
* Packet Generator
* WebGUI-Sniffer
* WLAN-Survey
* Router Discover
* WinShare Discover
* Remote Command & Control

### Livrables

À la fin de ce projet nous devons remettre au client, clé en main, le Raspberry Pi 2 configuré avec les différentes fonctions demandées implémentées, un mode d’emploi utilisateur ainsi qu’un manuel technique pour de futures extensions du produit.

### Maquettes préliminaires

Nous avons réalisé quelques maquettes des différentes interfaces lors du brainstorming de l’application.

![Affiche l'adresse IP, le Gateway ainsi que le netmask
Avec les boutons pour se rediriger sur les interface : Ping, trace et détail]()

Voici l’interface principale. Elle affiche les informations principales : l’adresse IPv4; l’adresse du routeur (gateway) ainsi que le masque de sous réseau.

La seconde interface, celle du ping, va nous permettre d’entrer l’adresse voulue puis de lancer le ping sur celle-ci. Un bouton va nous permettre de retourner sur l’interface principale (disponible sur toutes les interfaces).



L’interface des détails affichera les informations suivante : l’adresse IPv4; le masque de sous réseau, la passerelle par défaut, l’état du serveur DHCP et les adresses utilisées.



Le Traceroute, dernière interface que nous avions réalisé lors du brainstorming. Elle affichera le traceroute de l’adresse écrite.



## Planification

Pour la planification et le suivis du projet nous utilisons un fichier Google Sheet, mit à notre disposition pour le projet. N’étant pas assez complet, Monsieur Maréchal a demandé que chaque personne du groupe ait un journal de bord papier, à remplir lors de la réalisation du projet.

# 

# Conception

## Analyse de l’environnement

Pour réaliser ce projet, nous nous sommes penchés sur deux langages de programmation que nous n’apprenons pas durant les cours : le *Java* et le *Python*.

### Les logiciels utilisés

* **Eclipse IDE** : réalisation de l’interface et de quelques fonctions en *Java*.
* **Python** : IDE intégré sur le Raspberry Pi.
* **VNC Viewer** : accès à distance au Raspberry Pi.
* **Putty** : connection SSH au Raspberry Pi.
* **WireShark** : analyseur de paquets.
* **GitHub Desktop** : sauvegardes et versioning.

### Matériels utilisés

* **Raspberry Pi 2** : rends portable l’utilisation du logiciel.
* **Carte micro SD** : OS Raspbian installé.

### Langages utilisés

* **Java** : interface utilisateurs.
* **Python** : scripts permettant de récupérer les informations.
* **Bash** : script permettant de récupérer l’adresse IP.

### Sauvegardes et backups

Afin de faciliter le travail collaboratif ainsi que la réalisation de sauvegardes, il nous a été demandé d’utiliser un dépôt *Git*. Nous avons décidé de travailler avec *GitHub Desktop*. Le lien du dépôt est disponible ci-dessous :

* <https://github.com/snakeplayer/NSAei_software.git>

# Conception du code

### Java

Dans cette partie nous allons détailler les différents fichiers *.java* de notre projet *NSAei\_software* (ref. Annexe - Code Source).

***frmLaunch.java***

Cette classe est définie comme étant la configuration de lancement du projet lors du démarrage de l’application.

***frmMain.java***

Cette classe représente le première interface lorsque nous lançons l’application. Dans cette classe se trouve le code généré par JFrame pour créer les différents composant de l’interface.

***frmPing.java***

Cette classe représente l’interface du Ping, avec le code générant les différents composants de celui-ci.

***frmPingResult.java***

Cette classe représente l’interface du résultat du Ping lorsqu’on lance un ping depuis l’interface *frmPing.java*. Pour avoir un affichage cohérent nous avons mis la fonctions d’affichage en *Thread* ce qui nous permets l’affichage en même temps de l’arrivé du flux de réponse.

***frmTraceroute.java***

Cette classe représente l’interface du Traceroute, avec le code générant les différents composants de celui-ci.

***frmTracerouteResult.java***

Cette classe représente l’interface du résultat du Traceroute lorsqu’on lance un Traceroute depuis l’interface *frmTraceroute.java*. Pour avoir un affichage cohérent nous avons mis la fonctions d’affichage en *Thread* ce qui nous permets l’affichage en même temps de l’arrivé du flux de réponse.

***frmDetails.java***

Dans cette classe, représentant l’interface des détails, l'événement *Click* du bouton *Refresh* lance cinq scriptes : quatre écrit en *Python* qui permettent de récupérer le *Hostname*, *Gateway*, *Netmask* et la *MAC Address*; un en *Bash* qui va exécuter une commande pour récupérer l’adresse IP du Raspberry Pi.

### Python

***mac.py***

* Ce script permet de récupérer l'adresse MAC de l’appareil, à mettre dans le dossier de l'exécutable.

***hostname.py***

* Ce script permet de récupérer le nom de domaine de l’appareil, à mettre dans le dossier de l'exécutable.

***gateways.py***

* Ce script permet de récupérer la passerelle par défaut, à mettre dans le dossier de l'exécutable.

***netmask.py***

* Ce script permet de récupérer le masque de sous réseaux, à mettre dans le dossier de l'exécutable.

### Bash

# Ressources

### Librairies utilisées

* + 1. Python
* **Socket** : Récupération de l’interface réseaux.
* **Struct** : Convertis les struct de C en valeur Python.
* **Uuid** : Récupération de l'adresse MAC.
* **Fcntl** : Récupération du masque de sous réseaux.
  + 1. Java
* Le programme NSAei.
  + 1. Bash
* Récupérer l'adresse ip via script Bash.
* Lancer l’application au démarrage depuis “rc.local” et “.bashrc”.

# Conclusion

## Réalisation du projet

Lors du démarrage du projet, nous sommes parti dans l’idée de développer l’application en C#, étant donné que c’est le langage de programmation dans lequel nous sommes le plus à l’aise. Après avoir réalisé des recherches et différents tests, nous nous sommes rendu compte que cette technologie n’était pas la plus appropriée pour ce projet sur Raspbian.

Nous nous sommes donc penché sur une approche en Java. Nous avons réalisé quelques tests qui furent concluants, nous nous sommes alors lancé sur cette voie. Nous avons développé quelques unes des fonctions sur Windows 10 avec l’IDE Eclipse. Lors de l’import sur le Raspberry Pi, nous nous sommes rendu compte que certaines fonctions ne supportait pas le changement de plateforme. Pour y remédier, nous fait de nouvelles recherche pour trouver un autre moyen d’avoir accès au informations qui nous interessait. Nous avons donc utilisé des scripts en Python qui permettent de récupérer les informations voulues, exécutées depuis l’interface Java, afin de les afficher sur l’interface utilisateur.

L’attribution des tâches s’est effectuée plutôt naturellement. M. Troller, le plus à l’aise d’entre-nous avec le Raspberry Pi fut responsable de ce dernier : il s’est principalement occupé des différents scripts et du déploiement des nouvelles versions de l’outil. M. Schito et M. Chauche se sont plus concentrés sur le développement en Java ainsi que la rédaction de la documentation. Les échanges au sein du groupe furent très cordiaux et les compétences de chacun ont été pleinement mises à contribution.

## Résultat obtenu

Nous sommes très satisfait du résultat final. L’outil est fonctionnel, simple d’utilisation et globalement fiable.

## Améliorations possibles

Les différentes améliorations possibles à ce projet sont :

* Regrouper les différents scripts en Python et Bash dans un seul langage pour plus de cohérence.
* Au niveau du code, faire en sorte de séparer les différentes fonctions en classe Java.

Il faudrait également implémenter les fonctions demandées dans le cahier des charges que nous n’avons pu intégrer au projet, tels qu’un TFTP Server, DHCP Server, Packet Generator et un WinShare Discover.

# Annexes

* Code source (*NSAei\_JavaSources.pdf*)
* Configuration du Raspberry Pi (NSAei\_*Configuration\_RaspberryPi.pdf*)
* Manuel Utilisateur (*NSAei\_ManuelUtilisateur.pdf*)

1. http://www.schaffer-se.at/sseneu/produkte/nsa-der-network-status-analyzer/ [↑](#footnote-ref-0)