# Projet 1SIO janvier 2017
# STESIO - Analyse de logs



# Planning du 3 au 6 janvier

| | **Mardi** | **Mercredi** | **Jeudi** | **Vendredi** |
|---|---|---|---|---|
| **Matin (8h30 - 12h00)** | Projet | 10h - 12h : Film (Mme ARNAUD) | Projet | Projet |
| **Après-midi (13h30 17h)** | Projet | ½ journée à Nantes (Mme Engler - M. Grollier) | Projet | **Évaluation du projet** ***(évaluations croisées)*** |

# Cahier des charges

## Contexte

Pour être en conformité avec les obligations légales concernant la mise à disposition d'Internet, la société STESIO a mis en place un proxy pour journaliser les accès au web réalisés par ses salariés. A partir de ce journal, le responsable du système d'information (S.I.) souhaite établir des statistiques comme :

- les sites les plus visités
- la liste des utilisateurs les plus consommateurs

et dans les cas où cela est nécessaire (enquête de police par exemple) être capable de répondre à une requête du type :

- qui a consulté tel site, tel jour, à telle heure ?

Le fichier de log du proxy est un simple fichier texte (log_proxy.txt) contenant des informations sur les accès au web comme l'adresse IP, la date, l'heure, la commande HTTP utilisée (GET ou POST) , l'URL des différents éléments constituant la page téléchargée (images, bandeau, …). Ce journal étant d'une part, un fichier texte et d'autre part étant très volumineux, il est difficile à utiliser directement pour répondre facilement à ces besoins.

Le responsable du SI vous demande de créer une base de données sur ORACLE qui contiendra les tables suivantes :
SALARIES(num, nom, prenom, adresseIP) - clef primaire : num
PROXY(id, adresseIP, jourheure, URL) - clef primaire : id

***Précisions*** :

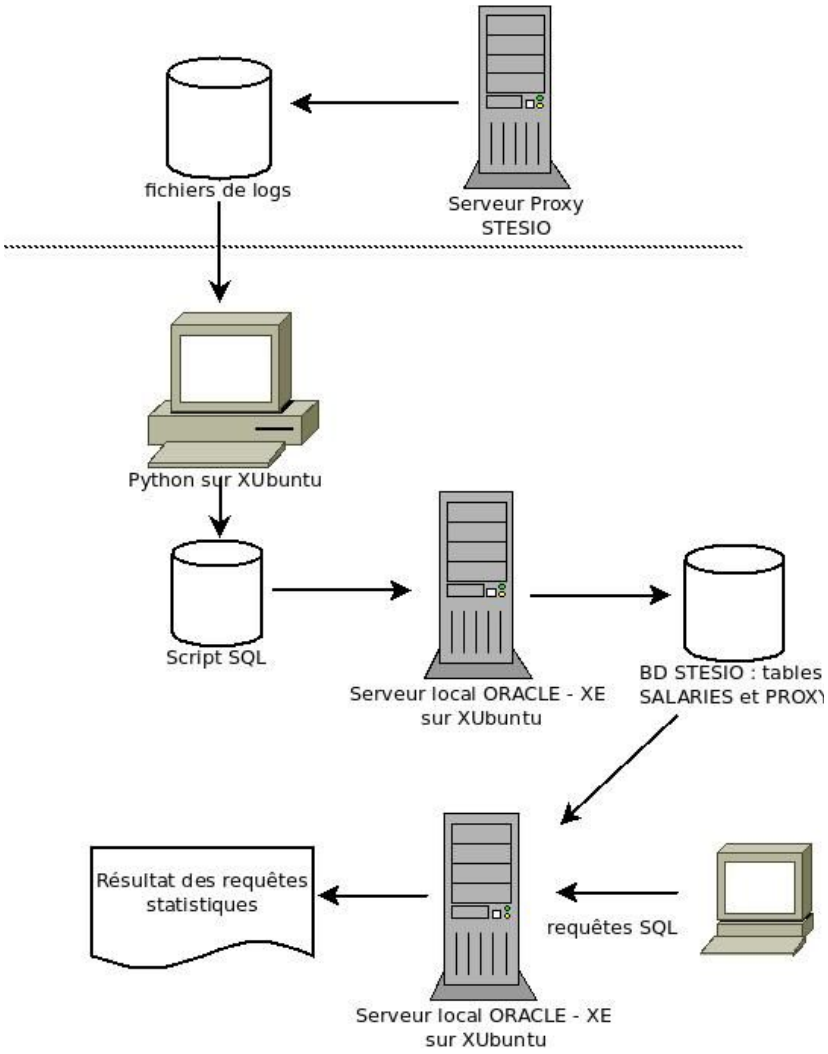
- *le champ adresseIP de la table SALARIES respecte une contrainte d'unicité*
- *le champ adresseIP de la table PROXY référence le champ adresseIP de la table SALARIES*
- *le champ id doit être auto-incrémenté (Cf. doc. jointe).*
- *le type DATE d'Oracle permet de mémoriser une date et une heure (Cf. doc. jointe) .*

Vous remplirez la table SALARIES avec les données suivantes :

| Num | Nom | Prenom | IP |
|---|---|---|---|
| 1 | DUPOND | Marie | 192.168.2.2 |
| 2 | DUBOIS | Paul | 192.168.2.1 |
| 3 | DURANT | Quentin | 192.168.2.3 |
| 4 | LEJAUNE | Laurence | 192.168.2.100 |

La table PROXY sera remplie par un script contenant des ordres SQL "INSERT INTO PROXY …" . Ce script sera généré par un programme, écrit en PYTHON, qui lira le fichier texte "log_proxy.txt" et qui, à partir des données lues, créera le script SQL.
Une fois le script généré, il suffira de l'exécuter dans ORACLE pour remplir la table PROXY. Il sera alors plus facile d'établir des statistiques.

***Processus de traitement***




## *Documents mis à disposition*

- le présent cahier des charges
- des jeux d'essai : les fichiers texte log_proxy_2017-01-01.txt, log_proxy_2017-01-02.txt, log_proxy_2017-01-03.txt
- deux documentations :
    - PYTHON : la gestion de fichiers et le traitement des chaînes de caractères
    - ORACLE : la gestion des dates et des identifiants auto-incrémentés
- la grille d'évaluation du projet qui sera utilisée lors de la recette finale.

## *Documents à produire*

A chaque étape, vous devez remettre des documents dans un fichier d'archive ZIP en respectant la convention de nommage suivante :

***nom1_nom2_etape_N.zip***

-> nom1 et nom2 sont les noms des deux étudiants du binôme ; N est le n° de l'étape.

Un travail a été ouvert à cet effet sur Moodle.

# Evaluation

Le vendredi, chaque binôme évaluera la réalisation de deux autres binômes : la grille d'évaluation est remplie en testant le jeu d'essai et le mode opératoire fournis par le binôme évalué.

# Les itérations du projet (étapes)

## **Étape 0** : Constitution des équipes

Vous devez constituer un binôme et le déclarer aux enseignants.

## **Étape 1**: Préparation de l'environnement de développement

Vous devez vérifier l'installation de Python sur la machine XUbuntu de développement (VirtualBox) et l'intégrer à votre environnement de développement NetBeans.

- Quelles sont les versions de python installées sur votre machine virtuelle XUnbuntu ?
- Installez un module additionnel de NetBeans pour pouvoir développer en Python :
  instructions : https://blogs.oracle.com/geertjan/entry/python_in_netbeans_ide_81
  téléchargement : http://plugins.netbeans.org/plugin/56795?show=true
- Sous Netbeans, créez un projet python "projetpython1sio" ; associez la version la plus récente de la plateforme python à ce projet.
- Recherchez un site de référence pour le langage python dans la version choisie. Donnez sont URL.

**A remettre** : un mode opératoire décrivant ces actions et comportant les réponses aux questions posées.

## **Étape 2**: Lecture et écriture dans un fichier texte en Python

Cette étape a pour but de vous former aux commandes Python permettant de lire et d'écrire dans un fichier texte(Cf. doc. Python).
Enregistrez les lignes suivantes dans un fichier "testEntree.txt" :

```
Emile;ZOLA
Victor;HUGO
George;SAND
```

Réalisez ensuite  un programme effectuant les actions suivantes :

- ouvrir le fichier testEntree.txt en lecture et un fichier testSortie.txt en écriture.
- pour chaque ligne du fichier testEntree.txt ::
  - extraire la première lettre du premier mot et le deuxième mot pour les concaténer
  - convertir la chaîne obtenue en minuscules
  - écrire cette chaîne dans le fichier testSortie.txt

**A remettre** :

- le code source Python commenté,
- le contenu des fichiers testEntree.txt et  testSortie.txt .

## **Étape 3**: Mise en place de la base de données en SQL

Vous devez créer un utilisateur (schéma) dans votre base de données ORACLE : nom STESIO, mot de passe STESIO.
Vous créerez les tables SALARIES et PROXY dans ce nouveau schéma, sans oublier les contraintes ni le contenu de la table SALARIES.
Donnez un exemple d'ordre SQL pour ajouter un enregistrement dans la table PROXY. Attention au format des dates pour Oracle. L'identifiant devra être auto-incrémenté (Cf. doc. Oracle).

**A remettre** : un document contenant ces ordres SQL, avec des explications.

## **Étape 4** : Analyse des fichiers textes de login

Vous analyserez les fichiers texte log_proxy_yyyy-mm-dd.txt : notez les informations disponibles sur chaque ligne et à quels champs de la base de données ils peuvent correspondre.

**A remettre** : un document décrivant votre analyse.

## **Étape 5** : Réalisation des requêtes de validation du jeu d'essai

Vous écrirez, sans les tester, les requêtes SQL correspondant aux **trois demandes** du cahier des charges :

1. la liste des sites les plus visités : nombre d'utilisateurs par site, en ordre décroissant
2. la liste des utilisateurs les plus consommateurs : nombre de lignes du fichier de log par utilisateur, en ordre décroissant
3. la liste des utilisateurs (nom, prénom, date et heure) ayant consulté un site, à une date donnée, entre telle heure et telle heure
   ***exemple** : qui a consulté le site de "l'équipe" le 05/01/2017 entre 7h02 et 7h03 ?*

**A remettre** : le code SQL des trois requêtes

## **Étape 6 :** Réalisation du programme Python de génération du script SQL INSERT

Vous réaliserez une première version du programme Python demandé. Ce programme lira la liste des enregistrements du proxy dans le fichier texte "log_proxy_yyyy-mm-dd.txt" pour générer un fichier de script SQL nommé "insert_yyyy-mm-dd.sql" contenant les commandes SQL nécessaires pour créer dans la table PROXY les enregistrements correspondant à la date "yyyy-mm-dd".

Dans un premier temps, l'utilisateur saisira uniquement la date du fichier de log au format "yyyy-mm-dd" (exemple : 2017-01-03). Le programme générera les noms des fichiers à partir de cette saisie.
Il sera ainsi possible de lancer le programme dans un terminal sous Linux de la manière suivante :

```
python3  nomDeVotreProgramme
```

**A remettre** :

- le code source PYTHON **commenté**
- les scripts SQL générés

## **Étape 7** : Exécution des scripts SQL d'insertion puis test des requêtes SQL

Exécutez les scripts SQL générés à l'étape précédente ; la table PROXY doit être remplie.
Vous testerez chacunes des trois requêtes conçues à l'étape 5.

**A remettre** : le compte-rendu du test, comprenant :

- le résultat de l'exécution des scripts d'insertion
- puis, pour chaque requête SQL :
    - le code SQL de la requête,
    - le résultat obtenu,
    - un commentaire.

## **Étape 8** : Rédaction d'un mode opératoire

Vous réaliserez à l'intention de l'administrateur système, un mode opératoire en français puis sa traduction en anglais ou en espagnol. Ce mode opératoire précisera :

- le format du fichier de log (structure)
- l'utilisation du programme Python
- l'utilisation des scripts SQL générés
- la procédure de test des requêtes de vérification des jeux d'essai

**A remettre** : le mode opératoire (bien) rédigé et mis en page, ainsi que sa traduction.

## **Étape 9 :** Première évolution de l'application

Mettre au point une nouvelle version du programme Python : seul le nom du fichier de log sera saisi par l'utilisateur, la date en sera extraite du nom du fichier par le programme.

> ***exemple** : l'utilisateur saisit le nom de fichier "log_proxy_2017-01-03.txt" ; la date extraite est "2017-01-03".*

Il sera ainsi possible de lancer le programme dans un terminal sous Linux de la manière suivante :

> ***python3*** *nomDeVotreProgramme*

**A remettre** :

- le code source Python **commenté**
- le compte rendu des tests effectués
- une nouvelle version du mode opératoire

## **Étape 10 :** Deuxième évolution de l'application

Mettre au point une nouvelle version du programme Python permettant de prendre en compte les paramètres en ligne de commande. Il sera ainsi possible de lancer le programme dans un terminal sous Linux de la manière suivante :

> ***python3*** *nomDeVotreProgramme nomFichierlog nomFichierSQLGenere*

**A remettre** :

- le code source Python **commenté**
- le compte rendu des tests effectués
- une nouvelle version du mode opératoire

## **Étape 11 :** Troisième évolution de l'application

Faire une nouvelle version du programme Python permettant de traiter plusieurs fichiers passés en paramètres.
***python3*** *nomDeVotreProgramme nomFichierlog_1 nomFichierlog_2 … nomFichierlog_n nomFichierSQLGenere*

**A remettre** :

- le code source Python **commenté**
- le compte rendu des tests effectués
- une nouvelle version du mode opératoire

## **Étape 12 :** Quatrième évolution de l'application

Mettre au point une nouvelle version du programme Python acceptant les caractères génériques (*, ?) dans les noms de fichiers. Exemple :
**python3** *nomDeVotreProgramme nomFichierlog_2017-01-*.txt nomFichierSQLGenere*

**A remettre** :

- le code source Python **commenté**
- le compte rendu des tests effectués
- une nouvelle version du mode opératoire

## **Étape 13 :** Cinquième évolution de l'application

Le serveur Oracle ne fonctionne plus. Vous n'avez plus accès aux tables PROXY et SALARIES. On vous demande en urgence de modifier votre programme Python afin qu'il puisse directement afficher le résultat des requêtes de l'étape 5 sans passer par SQL (toujours en partant des fichiers de log).
Il sera possible de lancer le programme dans un terminal sous Linux de la manière suivante :

***python3*** *nomDeVotreProgramme nomFichierlog*

**A remettre** :

- le code source Python **commenté**
- le compte rendu des tests effectués
- une nouvelle version du mode opératoire

## **Étape 14 :** Sixième évolution de l'application

Il est possible décrire des interfaces graphiques avec Python.
Pour cela, il faut importer une bibliothèque

- Tkinter en Python 2 *(attention : "T" majuscule en 1ère lettre)*
- tkinter en Python 3 *(attention : "t" minuscule en 1ère lettre)*

On peut gérer ainsi :

- des libellés
- des champs de saisie
- des boutons
- des menus

et bien entendu d'autres éléments d'une interface "digne de ce nom".

On vous demande de concevoir une petite interface graphique permettant de saisir directement les informations de la ligne de commande de l'étape 12, puis, via un bouton spécifique, de lancer le script.
Autant que possible, les messages prévus par votre programme de l'étape 12 apparaîtront dans l'interface (directement ou par l'ouverture de "pop-up").
Des boîtes de dialogues pourront également être prévues selon les besoins du programme (demande d'informations manquantes, rectification d'erreurs, confirmations, ...).
L'affichage d'une aide sommaire à la demande de l'utilisateur pourra éventuellement être prévu (via un bouton ou une option de menu).

**A remettre :**

- le code source PYTHON **commenté**
- le compte rendu des tests effectués
- une nouvelle version du mode opératoire, incluant le schéma ou la capture d'écran de la fenêtre principale de l'interface graphique ainsi que des pop-up et/ou des boîtes de dialogue diverses complémentaires, le tout disposé sur un schéma global avec des flèches de liaison montrant la logique globale de l'interface et quelques commentaires permettant à un non informaticien de la comprendre *(et de l'utiliser …)*.