P\_031 Serveur Web



David Dieperink, Stefan Petrovic

Alexis Rojas, Samuel Hörler

Classe : CID2a

ETML – N508

CP : Cyril Sokoloff

Tables des matières

[1 Introduction 3](#_Toc85014544)

[1.1 Titre 3](#_Toc85014545)

[1.2 Description 3](#_Toc85014546)

[1.3 Architecture virtuelle. 4](#_Toc85014547)

[2 Réalisation 4](#_Toc85014548)

[2.1 Analyse 4](#_Toc85014549)

[2.2 Journal de travail 4](#_Toc85014550)

[2.3 Arborescence du projet 5](#_Toc85014551)

[2.4 Description de l’Application et des Pages web 5](#_Toc85014552)

[2.5 Diagramme de communication http 7](#_Toc85014553)

[2.6 Configuration de l’ouverture de port du serveur 8](#_Toc85014554)

[2.7 Démonstration du bon fonctionnement 8](#_Toc85014555)

[2.8 Tests de connexion au serveur Web 10](#_Toc85014556)

[2.9 Contrôle d’écoute sur le serveur Web 11](#_Toc85014557)

[2.10 Contrôle de connexion au serveur Web 11](#_Toc85014558)

[3 Conclusion 12](#_Toc85014559)

[4 Trello 13](#_Toc85014560)

[5 Webographie 13](#_Toc85014561)

[6 Annexes 13](#_Toc85014562)

# Introduction

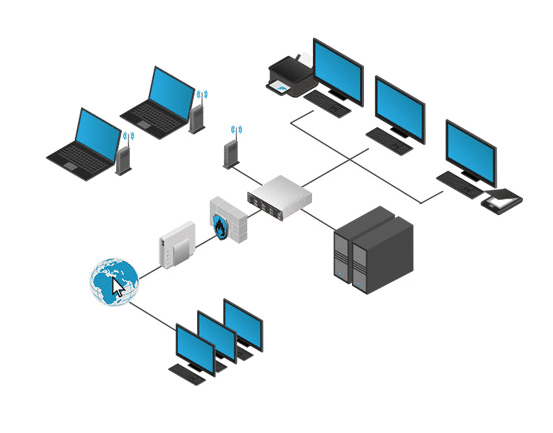
## Titre

Création d’un serveur web en console C#. Lançable et utilisable depuis 3 machines virtuelles.

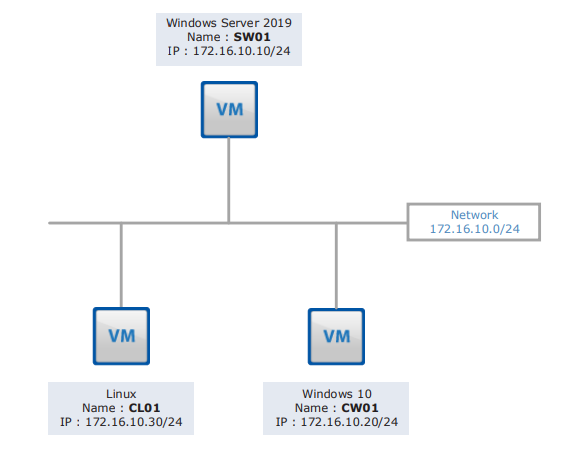
## Description

Ce projet consiste à développer un programme permettant de lancer un serveur web en local sur des machines virtuelles. Le programme est développé dans le langage C# en mode console. C’est-à-dire que les messages à transmettre au serveur seront envoyés en ligne de commande depuis la console.

Les 3 machines virtuelles sont différentes, tout d’abord il y’a la machine virtuelle qui nous sert de serveur. Elle est installée avec un Windows Serveur 2019. Ensuite les deux machines clients, une est installée avec un Windows 10 et la deuxième avec un Debian 10.9 (Linux).



## Architecture virtuelle.



# Réalisation

## Analyse

Tout d’abord nous avons analysé et lu le cahier des charges pour comprendre au mieux ce qu’il fallait faire. Une fois les choses comprises nous avons réfléchis à différente méthodes pour pouvoir gérer le projet à 4.

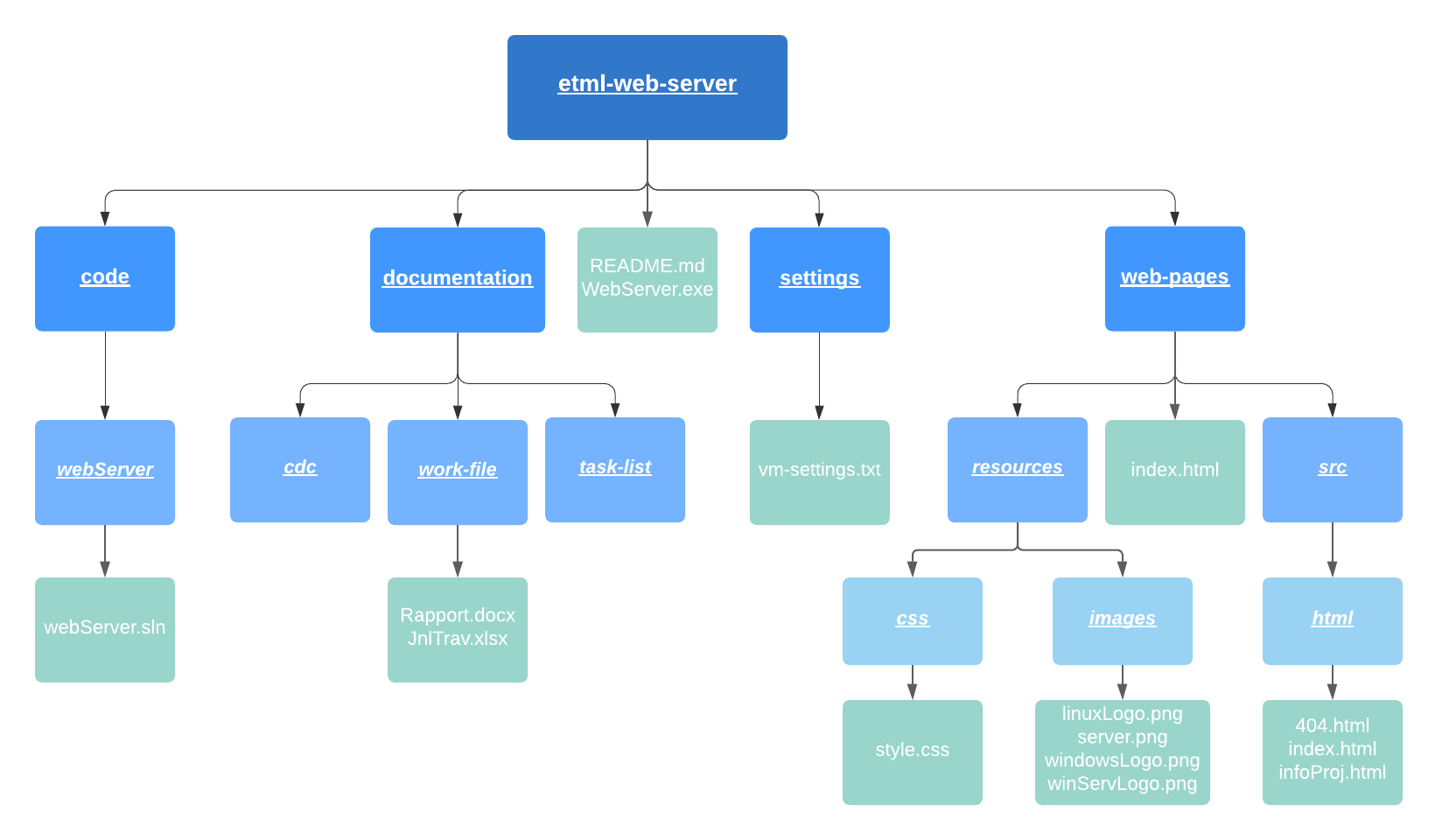
Nous avons fait le choix d’utiliser GitLab, c’est une plateforme en ligne qui est OpenSource est très appréciée des développeurs. L’avantage est que nous pouvons avoir une trace sur tous ce que nous avions fait lors du projet. De plus travaillé à 4 était un problème pour gérer les fichiers et qu’ils soient toujours à jour, donc GitLab était le moyen parfait pour résoudre ces soucis.

## Journal de travail

Dans le cahier des charges il était spécifier que nous devions faire un Journal de travail. On est parti sur l’optique de faire un seul journal de travail et de marquer ce que chaque personne à fait avec son prénom à côté.

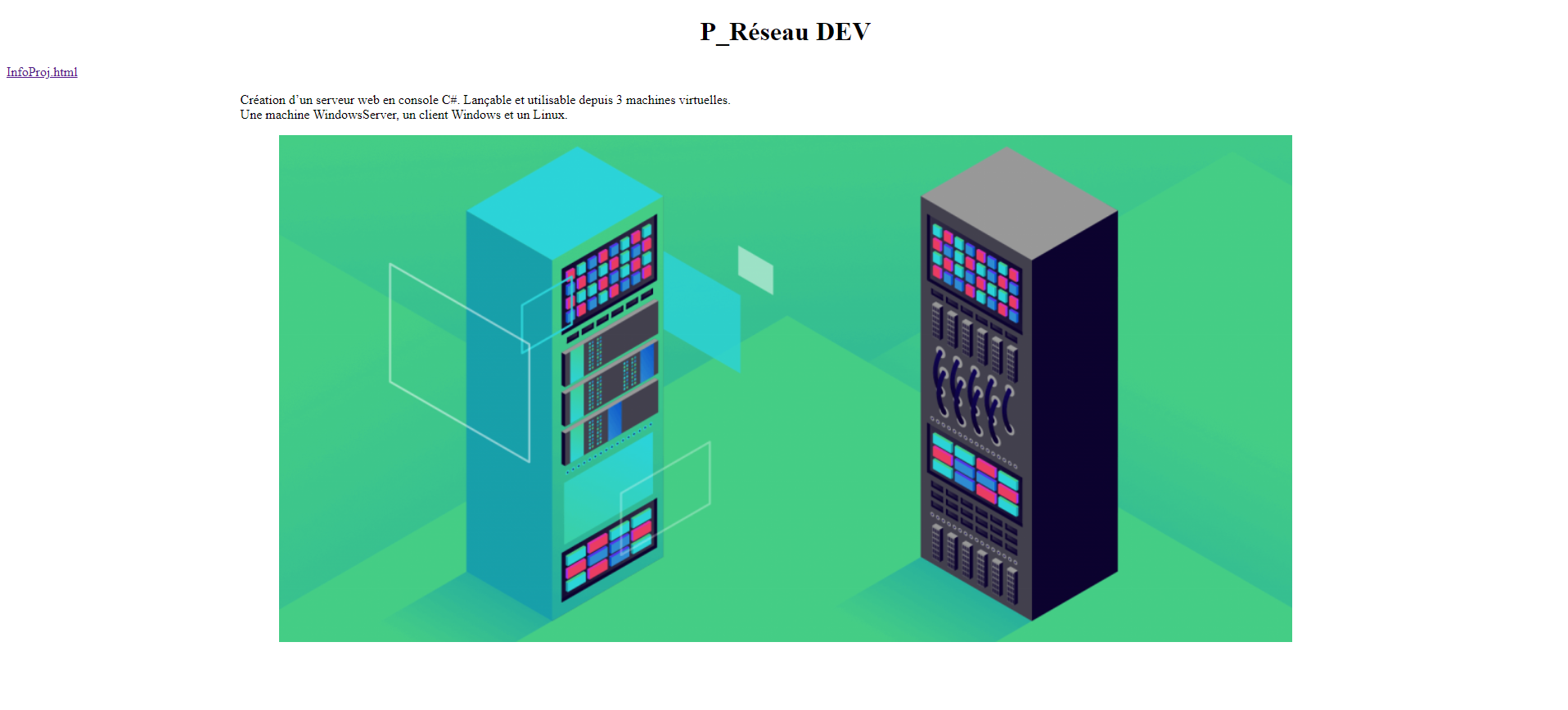
## Arborescence du projet

Voici une image représentant notre arborescence pour le projet :

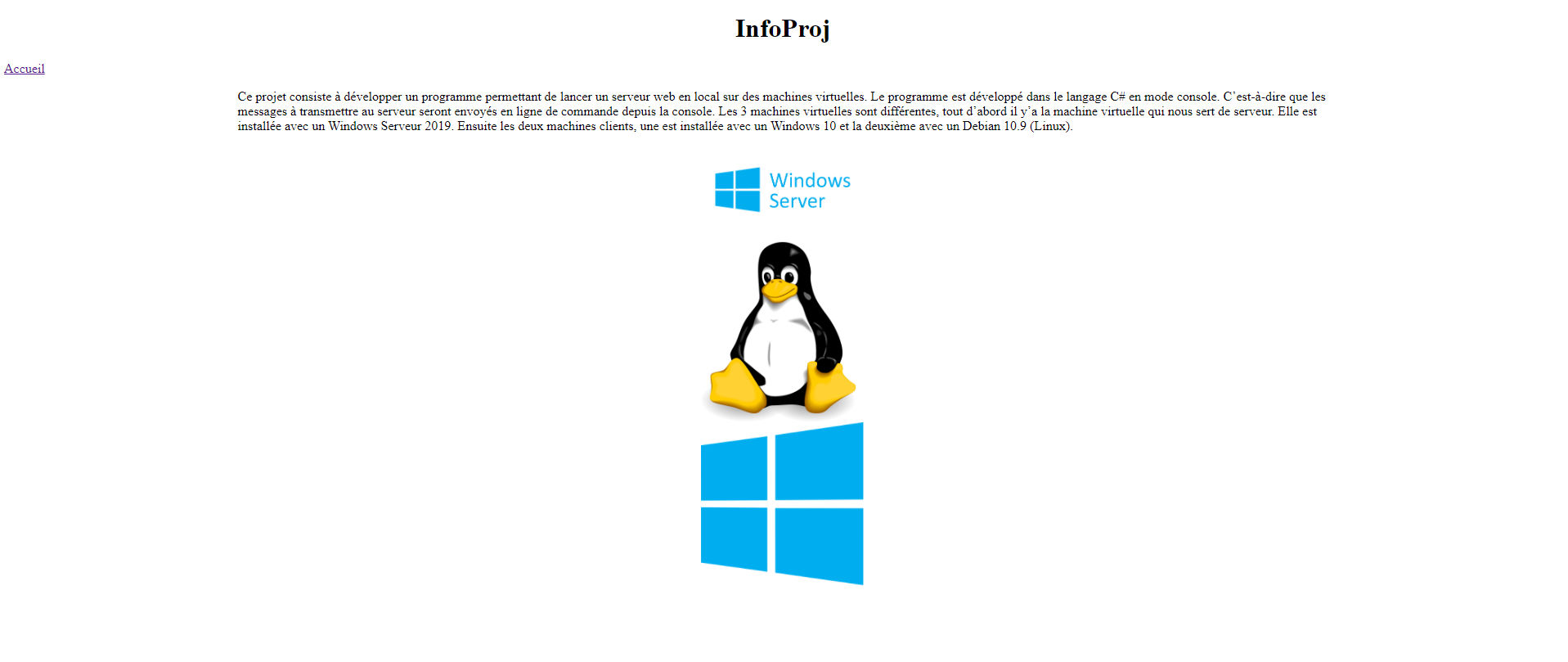


## Description de l’Application et des Pages web

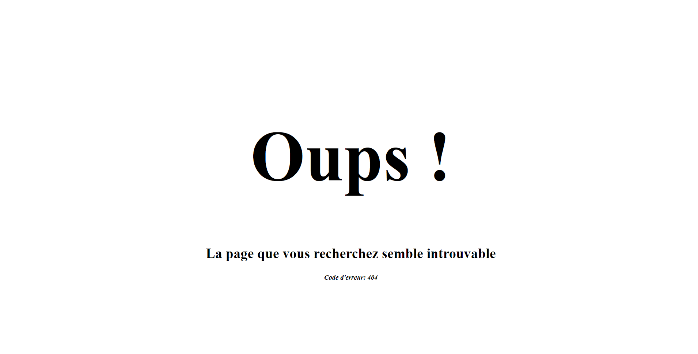
* Page d’accueil : Voici la page d’accueil ou il y a un petit descriptif du projet « Ce projet consiste à faire un serveur web en console C# pour des machines virtuelles. Dans ce projet nous avons dû faire un site composé de 3 pages, des machines virtuelles et un serveur web en console ».



* Page descriptive du projet : Ceci est la page d’information du projet c’est ici que sera décrit plus précisément le projet comme par exemples : le système d’exploitation des machines, la configuration du serveur, le langage utilisé pour le site et pour le serveur web.

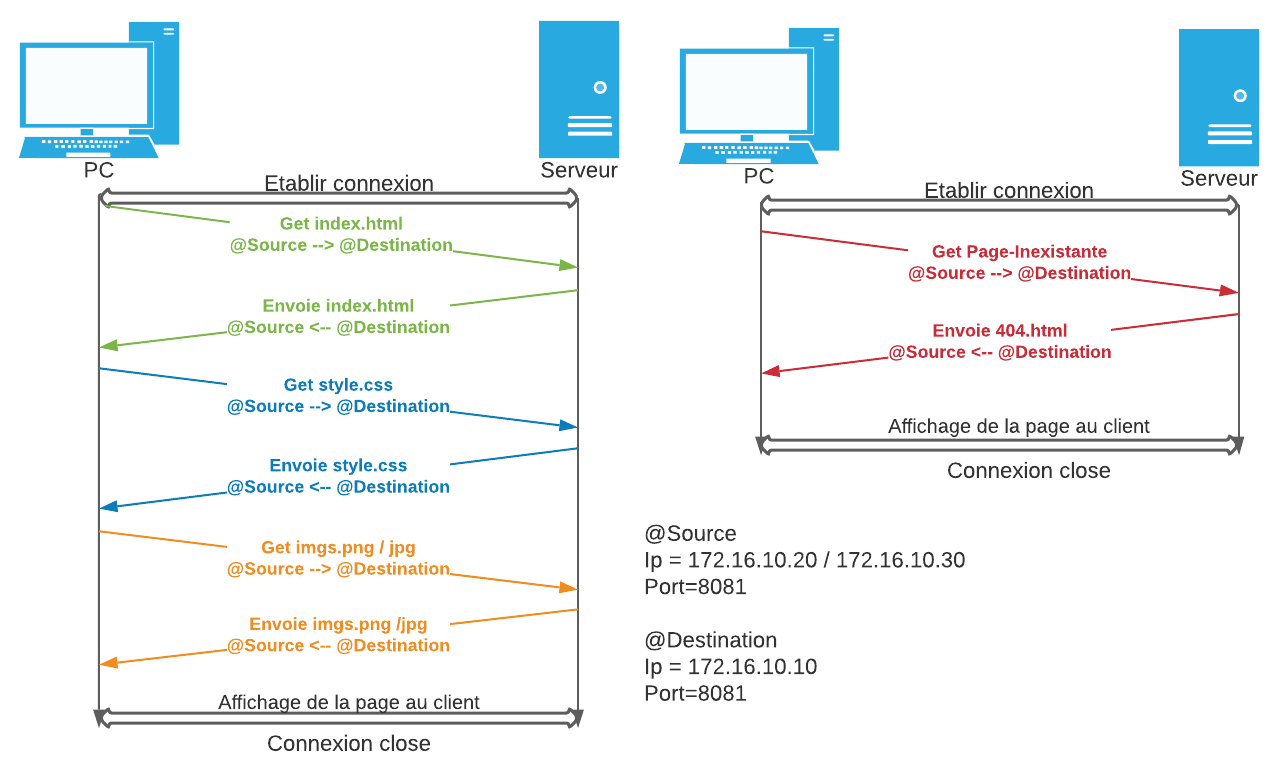


* Page d’erreur 404 : Il s’agit de la page erreur 404 qui s’affichera lorsque le client recherchera une page non existante.



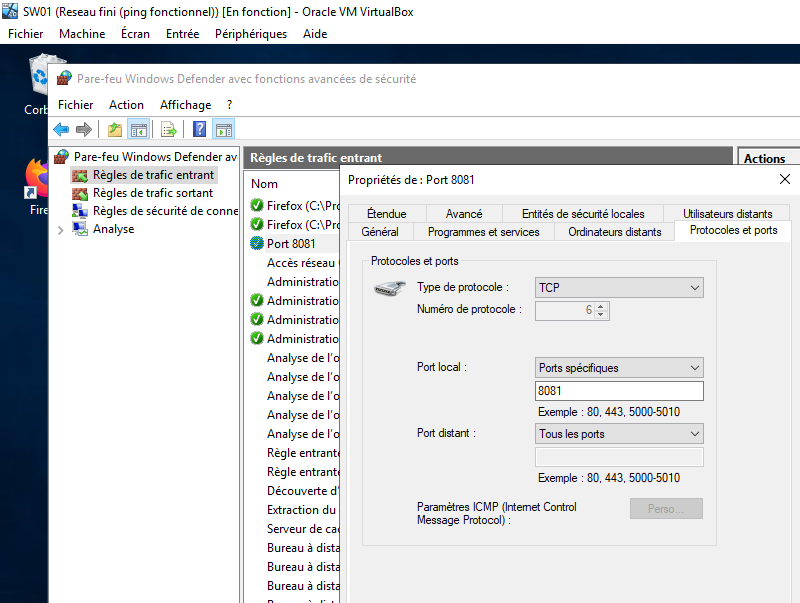
## Diagramme de communication http

* A gauche, le diagramme de communication lorsque le client essaie d’accéder à une page web existante sur le serveur.  
  Le client fait la requête et le serveur lui envoie tous les fichiers retrouvés à l’intérieur de la page web (le style CSS, les images, des scripts JS, etc…), puis la page web s’affiche sur le navigateur.
* A droite on peut voir le diagramme de communication lorsque le client essaie d’atteindre une page qui n’existe pas dans le serveur.  
  Le serveur va vérifier l’existence de la page recherchée et s’il ne la trouve pas il envoie la page d’erreur 404.



## Configuration de l’ouverture de port du serveur

* Ajout d’une règle entrant avec le protocole **TCP** et le port local **8081.**



## Démonstration du bon fonctionnement

* Accès page d’accueil depuis le client Windows (figure 1 : Client Windows, figure 2 : Serveur Windows2019) :

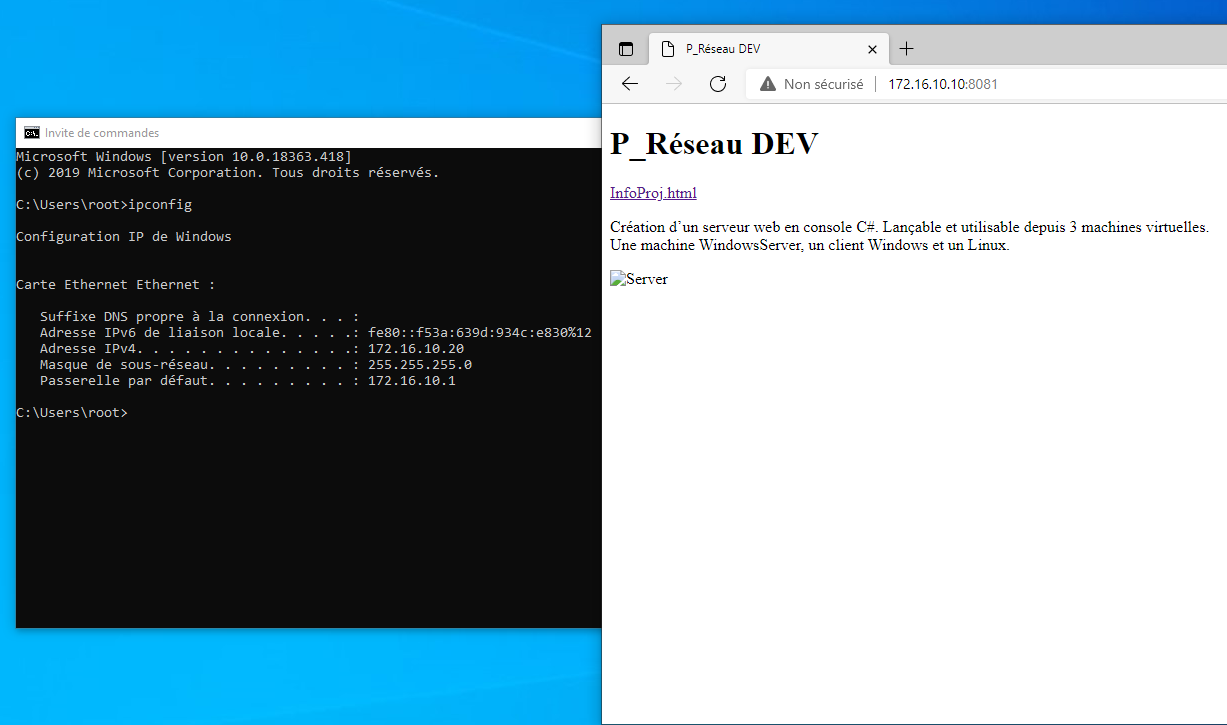


Figure 1 Client Windows

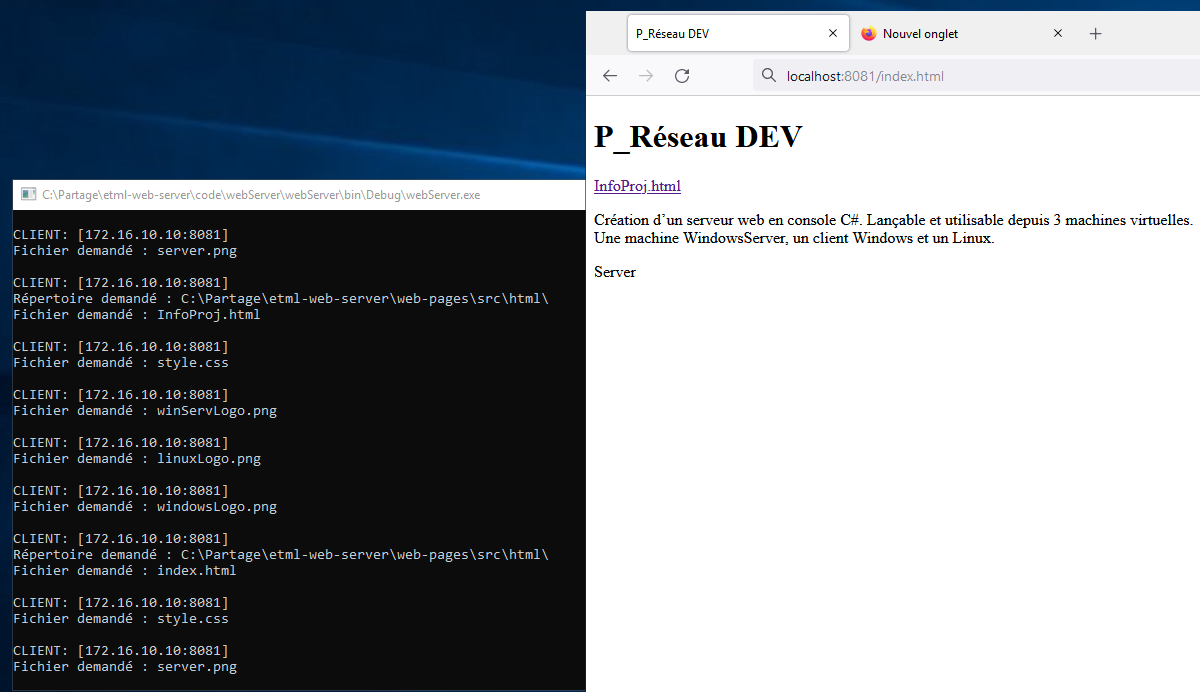
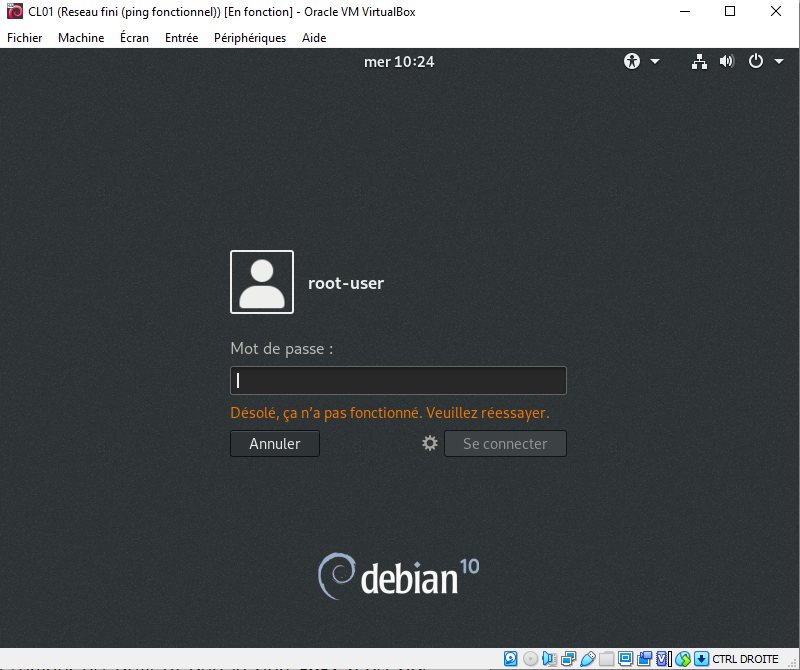


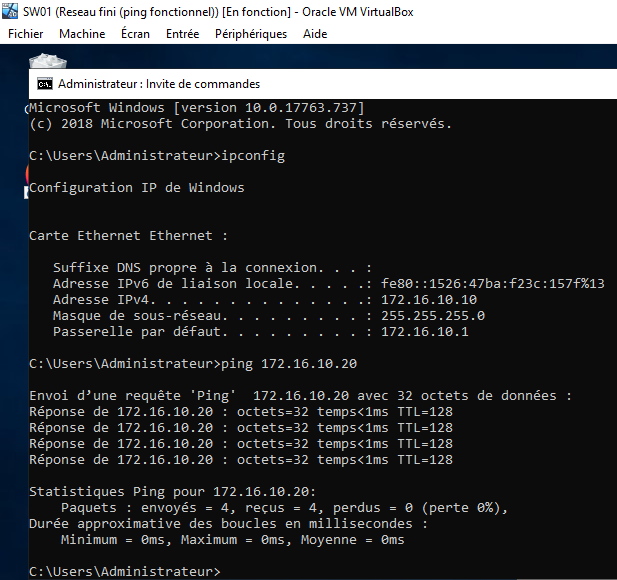
Figure 2 Serveur Windows2019

* Accès page d’accueil depuis le client Debian – Linux (Erreur de connexion à la machine virtuelle lors des dernières minutes, donc pas possible de tester).

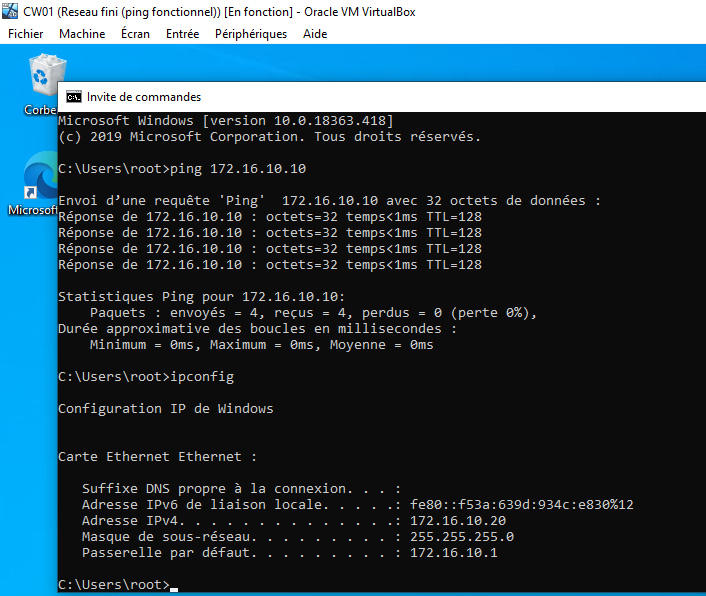


## Tests de connexion au serveur Web

* Ping depuis le serveur (**SW01** :172.16.10.10) vers le client Windows (**CW01** :172.16.10.20)



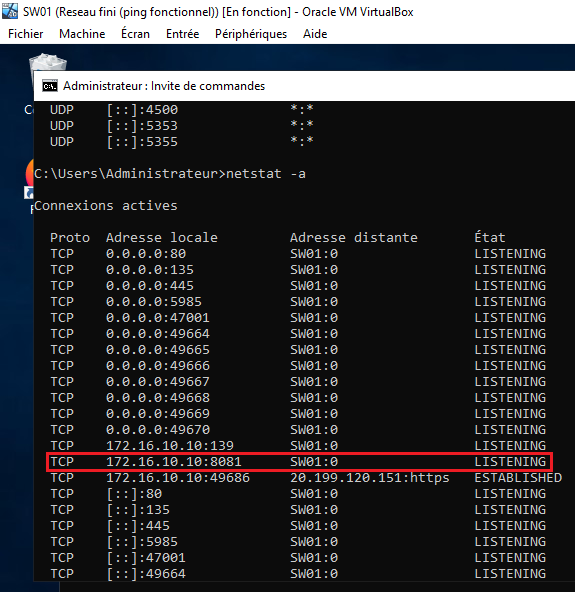
* Ping depuis le client Windows (**CW01** :172.16.10.20) vers le serveur (**SW01** :172.16.10.10).



* Connexion entre le Serveur web (**SW01**) et le client Linux (**CL01**) pas possible à cause d’un problème de connexion sur le client lors des dernières minutes.

## Contrôle d’écoute sur le serveur Web

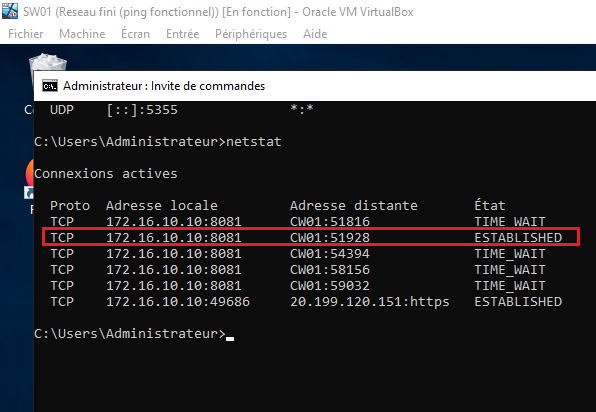
* Serveur web en écoute sur le port **8081**(NetStat) pour le client Windows (**CW01**) :



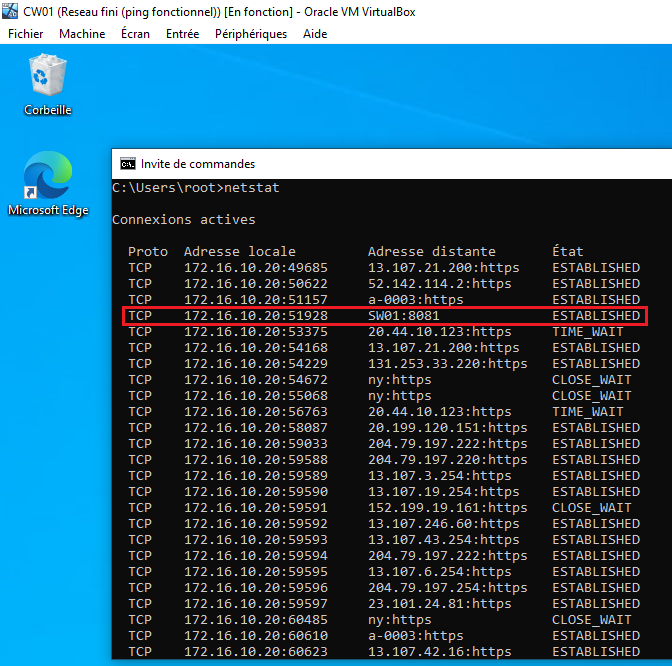
* Contrôle d’écoute entre le Serveur web (**SW01**) et le client Linux (**CL01**) pas possible à cause d’un problème de connexion sur le client lors des dernières minutes.

## Contrôle de connexion au serveur Web

* Connexion établie entre le serveur et le client Windows (CW01).



* Connexion établie entre le client Windows (**CW01**) et le serveur (**SW01**).



* Connexion entre le Serveur web (**SW01**) et le client Linux (**CL01**) pas possible à cause d’un problème de connexion sur le client lors des dernières minutes.

# Conclusion

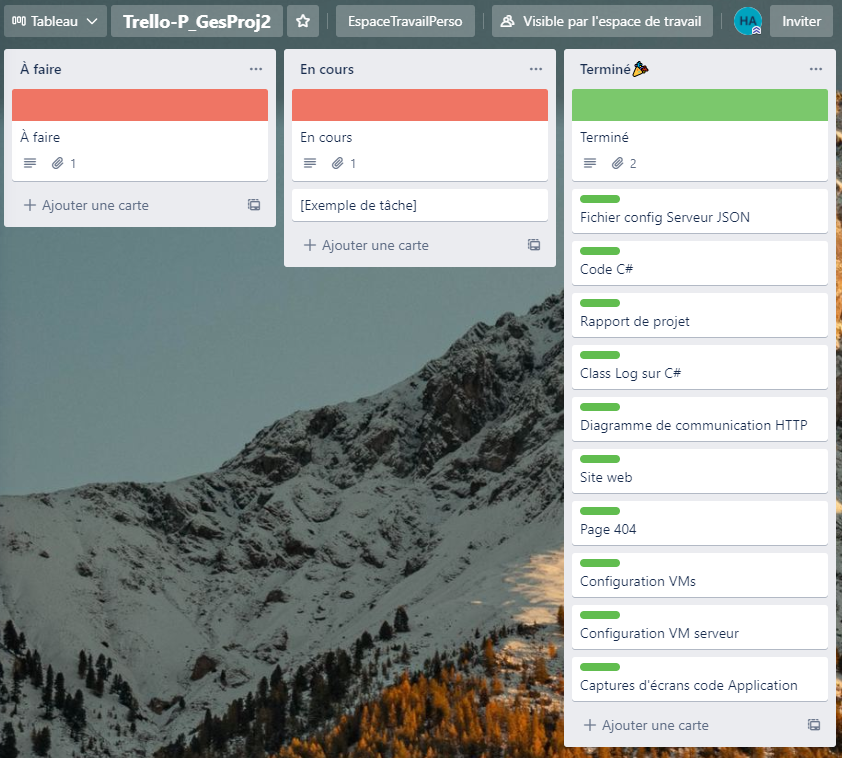
Le serveur web C# permet l’accès aux pages web depuis une autre machine connectée au même réseau, cependant l’application n’arrive pas à envoyer les styles CSS ni les images qui se trouvent à l’intérieur de chaque fichier Html.

L’ouverture des portes et la communication entre les clients et le serveur est correcte donc, on a pu conclure que l’erreur se trouve sur le code source de l’application.

La journalisation du serveur web fonctionne parfaitement puisque le fichier log enregistre tous les événements de connexion. Enfin le fichier « config.json » permet de paramétrer l’IP et le port d’écoute du serveur web.

# Trello

* Voici le Trello avec toutes les tâches accomplies.



# Webographie

* [Creating your own Web Server using C# (c-sharpcorner.com)](https://www.c-sharpcorner.com/article/creating-your-own-web-server-using-C-Sharp/)
* [Procédure : ouvrir un fichier journal et y ajouter des éléments | Microsoft Docs](https://docs.microsoft.com/fr-fr/dotnet/standard/io/how-to-open-and-append-to-a-log-file)
* <https://www.technical-recipes.com/2016/creating-a-web-server-in-c/>

# Annexes

* Journal de travail de groupe
* Trello contenant les tâches du projet
* Pages Web