|  |  |
| --- | --- |
|  | Project-ZIA |
|  | Résultat de recherche d'images pour "image serveur" |
| ZIA | Le serveur http |
|  | Le zia est un projet scolaire effectué au sein de l’école EPITECH Lyon au cours de notre troisième année. Réaliser en groupe ce projet a pour but de récréer un serveur http cross-plateforme.  Auteur : Junique Virgile, Maxime Maisonnas, Quentin Albertone, Rodolphe Chazal, Jeremy Koehler, Loris Sanchez. |

Project-ZIA

Le serveur http

**SOMMAIRE :**

1. Présentation du projet 2.
2. Ses technologies 3.
3. Organisation 4.
4. Sa structure 5-11.
5. Ses fonctionnalités 12.
6. Conclusion 13.

**Présentation :**

Au cours de notre troisième année dans l’école EPITECH, il nous est donnée la possibilité de réaliser en groupe le projet ZIA dans le module c++/httpd, dans lequel nous devons reproduire un serveur http.

Celà consiste à créer dans une première partie une API pour ce type de serveur qui sera utilisée par tous les groupes de l’école afin d’avoir une même base. Cette API sera ensuite soumise à un vote et peut-être sélectionnée parmi toutes les autres. Puis dans un second temps, de réaliser le développement de ce projet.

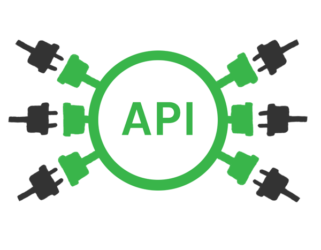
**Mais qu’est-ce qu’un serveur http :**

Selon Wikipédia un serveur http est : « Un serveur HTTP ou daemon HTTP ou HTTPd ou serveur Web est un logiciel servant des requêtes respectant le protocole de communication client-serveur HyperText Transfer Protocol (HTTP), qui a été développé pour le World Wide Web. »

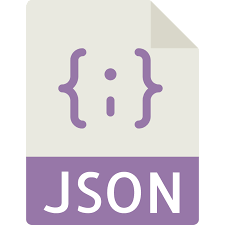
Une définition bien compliquée, pour simplifier, http est un protocole de communication entre le site internet que vous visitez et vous, permettant ainsi de vous communiquer les données de ce site via un système de requêtes.

**Mais qu’est-ce qu’une API :**

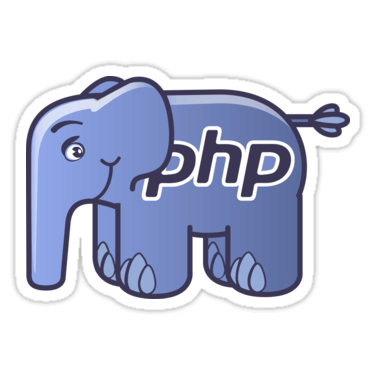
C’est un ensemble normalisé de classes, de méthodes ou de fonctions qui sert de façade pour un logiciel lui permettant d’offrir ses services à d'autres logiciels.

**Ses Technologies :**

Pour ce faire le ZIA est un ainsi muni de :

* Son api (Simple Zia API : https://github.com/kiwixz/sza)
* Du JavaScript Object Notation (JSON) pour sa configuration.

C’est un format de fichier permettant de représenter

les données de manière structurée dans un fichier texte.

* De plus le zia doit être fourni avec les modules SSL et PHP.

Pour résumer le module SSL permet une connexion sécurisée et le module PHP l’affichage de sites internet dynamiques.



* Enfin le serveur est développé

à l’aide du langage C++.

C’est un langage de programmation orientée

objet créé en 1983 permettant la création de tout

type de logiciel.

**Organisation :**

Notre groupe composé de : Maxime Maisonnas, Virgile Junique, Loris Sanchez, Quentin Albertone, Rodolphe Chazal et Jérémy Koehler.

Nous avons utilisé un Trello (Application Web permettant la création de tableau de tache) pour se répartir le travail ainsi que plusieurs branches sur notre Git (système de dossier personnaliser et partager à distance). Pour ainsi se structurer de la manière suivante :

**Sa structure :**

**Getting started**

Le serveur open-source ZIA est disponible uniquement sur Linux, cependant l’installation s’effectue seulement via le gestionnaire de package pacman en passant par le repository officiel. Sur Windows ouvrez le dossier cpp\_zia, gênerez et exécutez zia.exe via Cmake.

Son utilisation intuitive et performante en fait un parfait allié pour vos futurs projets. Parfaitement modulable, n’hésitez pas à apporter votre contribution pour intégrer de nouveaux modules ou tout simplement continuer à l’améliorer !

**Installation :**

Cloner le repository officiel :

git clone git@git.epitech.eu:/maison\_m/cpp\_zia

Compiler le projet :

cmake . && make zia

Rajouter les lignes suivantes, en remplaçant path/to par le chemin où vous avez cloné le repository, à la fin de votre /etc/pacman.conf :

*[zia]*

*SigLevel = Never*

*Server = file:///path/to/cpp\_zia/linux/package*

Taper les commandes suivantes dans votre terminal :

sudo pacman -Syu && sudo pacman -S zia

**Désinstallation :**

Taper simplement la commande :

sudo pacman -Rsn zia

Retirer les 3 lignes concernant le paquet zia dans votre /etc/pacman.conf pour ne pas être importuné lors des prochaines mises à jour.

**Usage :**

Notre zia dispose d’une page de manuel disponible avec la ligne de commande suivante : man ./linux/packge/zia.8.gz OU man zia

OPTIONS :

-c, --config=file Use an alternative configuration file instead of /etc/zia/zia.json.

-r, --reload Tell master process to reload its configuration file.

-s, --signal=sig Send signal sig to the master process. The argument sig can be one

of: stop, quit, restart, reload. The following table shows the corresponding system signals :

Stop SIGTERM

Quit SIGQUIT

Restart SIGHUP

Reload SIGUSR1

-h, --help display help and exit.

EXIT STATUS :

0 if OK

1 on failure

2 if permission denied

EXAMPLES :

Lancer le zia avec un fichier de configuration précis :

zia -c ~/zia.json

Recharger la configuration du serveur :

zia -r

Arrêter le processus principal :

zia -s stop

**Configuration :**

Pour configurer votre serveur zia vous disposez d’un fichier par défaut situé : /etc/zia/zia.json

Cependant s’il n’existe pas il sera : $(Repository)/conf/zia.json

Il vous est toujours donné la possibilité de lancer le serveur en spécifiant le chemin du fichier de configuration via l’option -c (cf : man zia).

Ce sont des fichiers au format JSON ([www.json.org/json.fr](http://www.json.org/json.fr)) Vous pouvez disposer des clés et valeurs que vous désirez pour vos modules personnels ou changer les valeurs par défaut du service http (port, max\_uri\_size, ...). Vous trouvez une description des valeurs par défauts dans le manuel.

Pour charge un/des module(s) le fichier de configuration doit contenir :

- "modules" : Liste des noms des modules ex : "ssl", "php" (sur linux vos modules doivent être nommés de la manière suivante : "lib" + module name + .so. Dans le fichier renseignez uniquement : module name)

- "modules\_path" : Chemin d’accès aux modules ex : "zia/modules", "/bin/modules"

Une fois le serveur lancé, la configuration est chargée dans un objet de type ConfObject en respect avec l’API imposée.

# **Daemon :**

Notre exécutable zia permet de lancer notre serveur ou de communiquer avec le processus déjà existant.

Lorsque l’utilisateur démarre le serveur, le daemon vérifie que les prérequis pour le lancement du zia sont corrects. Il est le seul moyen de communication entre l’utilisateur et notre serveur. Il stocke son pid dans le fichier */run/zia.pid* et intercepte les signaux réalisant des actions définies :

|  |  |
| --- | --- |
| SIGTERM, SIGINT, SIGKILL | Forcent l’arrêt du serveur |
| SIGQUIT | Éteint le serveur proprement |
| SIGHUP | Redémarre le serveur |
| SIGUSR1. SIGUSR2 | Rechargent les fichiers de configuration |

Lancer notre serveur en arrière-plan nous permet de laisser un plus grand nombre de TTY à disposition de l’utilisateur. Les sorties standard et d’erreur sont désactivées laissant place aux file descriptors des sockets et des clients. Il est ainsi compatible avec les gestionnaires de services. L’utilisateur peut déléguer la gestion de notre zia à linux qui le lancera à chaque démarrage.

Ce lancement à pour effet de charger les modules, vérifier ceux encore en cours de fonctionnement et ainsi de lancer le processus chef que l’on nommera NetStack par soucis de clarté. Il sera la clé de voute de notre zia.

**NetStack :**

Le NetStack du serveur Zia contient quatre parties :

Gestion des Utilisateurs - Communication avec le client

C’est l’interface qui va gérer les utilisateurs. Cela compte les connections et les déconnections, les pertes de clients (lors d’erreurs côté client) et les envoie de requêtes. C’est le point d’entrée du serveur.

Gestion des Requêtes - Communication avec le serveur

Complètement indépendant des états des client cette partie recense les requêtes faites par les utilisateurs. Elle lie les requêtes avec les utilisateurs et leurs historiques de connexion.

LoadBalancer - Communication inter-process

Le LoadBalancer va ensuite communiquer avec les différents processus qu’il aura lancé au démarrage afin de céder le point d’entré lié au client et à sa requête (= le fd du client).

Workers - Communication avec le service http

Les workers sont des processus différents (nouveau PID) de la partie réseau du serveur. Ils lisent sur la socket Unix (communication interne au serveur) les FDs des clients leurs permettant de lire les requêtes des clients. Ils répondent directement au client, ce qui permet de ne pas encombrer le NetStack.

Cette implémentation permet la parallélisation de tâches simples, et ce pour aller plus vite dans la gestion des requêtes. Cela permet aussi aux actions importantes de s’exécuter sans avoir d’incidence sur le reste du serveur. Si un worker est supprimé ou ne répond plus, l’intégrité du serveur n’est pas compromise pour autant. Ils permettent aussi lors de l’arrêt prématuré du NetStack de finir la requête en cours.

Les moyens de communication du serveur sont les sockets INETs pour les communications avec les clients et UNIX pour les échanges interne.

**Gestion de modules :**

Au démarrage les modules renseignés dans le fichier de configuration sont chargés et stockés en respect avec l’API (SZA).

Il est responsable de l’ajout, suppression ou du rechargement de modules. L’API dispose de deux méthodes clés : Exec, Conf permettant d’interagir avec les modules et de les configurer.

Le gestionnaire de modules est directement intégré dans le NetStack permettant d’interagir avec tous les modules chargés et d’exécuter leurs services selon un ordre de priorités.

**Modules :**

Notre zia possède deux modules à dispositions de l’utilisateur. Que l’on peut trouver dans le dossier modules de notre zia.

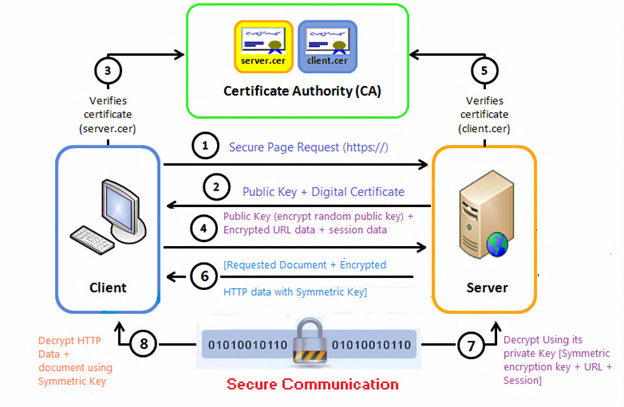
Le module SSL permet le traitement de requête https ainsi proposant une connexion sécurisée et certifiée.

Le module PHP qui permet la gestion du PHP.

Le zia dispose aussi d’un système de Log indépendant des modules qui permet d’afficher les messages du serveur et de les stocker dans un fichier de log, situé au chemin suivant : /var/log/zia/zia.log

* **SSL :** (secure socket layer)

Afin d’établir une connexion sécurisée nous avons un module SSL utilisant la librairie Openssl. La connexion sécurisée est initialisée par une étape appelée “Handshake”. Une fois le handshake réalisé, les clés échangées, et les certificats vérifiés, client et serveur communiquent de manière sécurisée en cryptant/décryptant les données.



Il se configure ensuite comme cela :

|  |  |
| --- | --- |
| {    "SSL" : {      "cert" :"CertName",      "key" : "KeyName"   }  } | CertName doit être remplacé par le chemin permettant d’accéder au certificat.  KeyName doit être remplacé par le chemin permettant d’accéder au certificat.  Pensez à rajoutez la clé modules\_path |

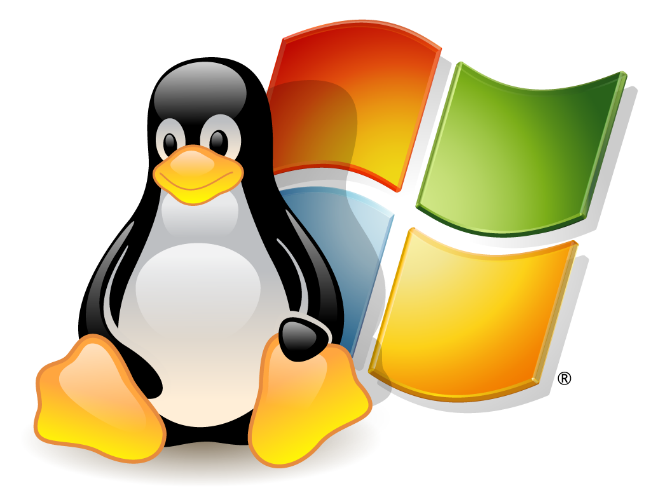
* **PHP :** (Hypertext Preprocessor)

Pour la création de contenu dynamique, la gestion de requête de type GET, POST, … il a été nécessaire de développer un module PHP sous le format CGI (common gateway interface) permettant d’interpréter les demandes que génère une requête PHP, via l’usage de la librairie php-cgi. Il ne nécessite aucune configuration à part de figurer dans la liste des modules.

**Ses Fonctionnalités :**

* Notre ZIA permet s’exécuter en daemon, c’est-à-dire qu’il tourne en tache de fond sur la machine sur laquelle il est installé.
* Il se configure à l’aide de fichier JSON puissant et simple

D’utilisation permettant à l’utilisateur de facilement le configurer.

* Il est possible de compiler et de lancer notre zia sur les distributions UNIX et Windows. Bien que pour le moment toutes les fonctionnalités de notre serveur ne sont pas opérationnelles sur WINDOWS.
* Ce serveur propose une connexion sécurisée via notre module personnel SSL.
* De plus l’hébergement de site internet dynamique y est possible à l’aide de notre module PHP-CGI pouvant intégrer par exemple une base de données.
* Enfin l’utilisation et la gestion du protocole de communication

Http y est complète et RFC compliant avec les plus

Importants logiciels utilisant ce protocole.

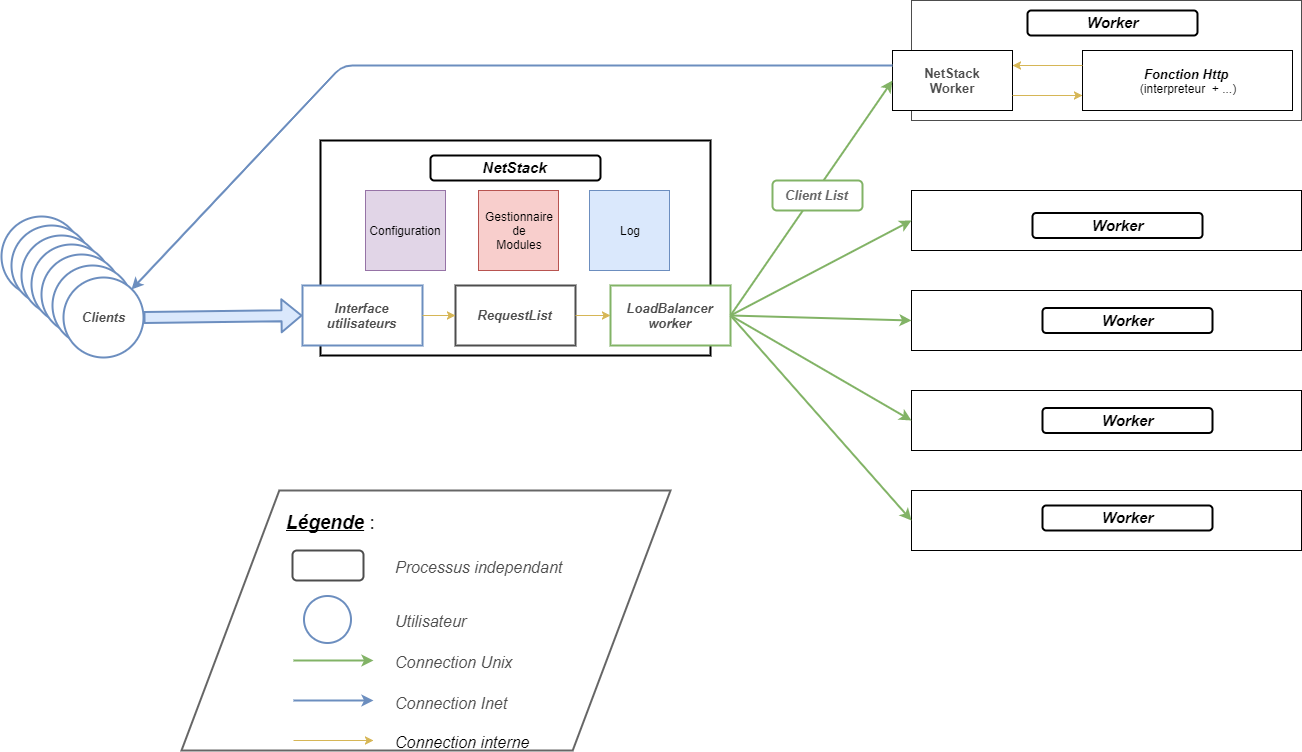
**Conclusion :**

Le projet ZIA fût long et parfois semé d’embuches mais ce fut de manière globale très intéressant de réaliser ce projet et d’en apprendre d’avantages sur le fonctionnement de ce type de serveur. Bien que nous utilisions ce genre de serveur au quotidien et souvent sans même les comprendre, voir et approfondir nos connaissances a été très enrichissant pour nous.

De manière plus technique notre projet est fonctionnel sur les systèmes d’exploitation de type UNIX. Malgré un manque de temps et une structure ambitieuse nous n’avons pas pu réaliser l’adaptation de notre code avec le système WINDOWS. En effet les appels systèmes n’étant pas les mêmes nous devions dans un dernier temps créer une version adaptée au système WINDOWS.

Néanmoins nous jouissons d’un serveur complétement opérationnel nous offrant la possibilité d’hébergés des sites internet avec un système de log adapté pour l’administrateur du serveur.

Enfin la réalisation de ce projet clôture notre troisième d’année d’étude à l’école EPITECH Lyon.

**Annexe :**