

RAPPORT

Refont de l'application

Par
Ronald PINA GUILLEN

Table of Contents

I - Présentation général	3
II - Descriptive technique.....	4
III - Maintenance application.....	5
a) Réorganisation des répertoires et fichiers	5
b)Nomenclature du code	11
1 – Context.....	11
2 – Résultat	13
c) Re-factorisation du code source	14
.....	14
1 – Décortication du code	15
2 - Résultat de la séparation :.....	17
3 - Test sur le navigateur web :.....	17
4 – Changement de l’architecture	18
IV – Gestion des panneaux	23

I - Présentation général

Archéan Technologie est une entreprise qui s'occupe, principalement, du sonore-sécurisation dans les ERP (Établissement recevant du public), les infrastructures liées aux transports et les sites à risque.

Ils produisent des produits de la gamme VoIP comme un contrôleur de réseau et de processus (permet de superviser et de gérer en temps réel le système de sonorisation), pupitre tactile (permet d'initialiser une action sur le système de sonorisation : activer une diffusion, une annonce et etc.), et d'autre produits.

Monsieur Hugo Rodrigues, le responsable de développement informatique, a conçu des applications web qui sert à gérer le système de certain produit comme ATCONTR2 et ATTOUCH. Il a perçu que ces applications web contiennent un quelque structure ou fonctionnalité qui sont les mêmes et donc, il veut centraliser ces applications dans une seul application où il peut éventuellement gérer les panneaux selon le produit, c'est à dire, de pouvoir désactiver ou activer les panneaux.

Il me donne comme mission :

- De réorganiser tous les codes sources de chaque ancienne application à l'intérieur de l'emplacement de la nouvelle application, où chaque fonctionnalité peut être retrouver dans leur propre répertoire.

- De décortiquer certains codes et placez-les dans leurs répertoires respectifs, aussi refactorisé certain codes pour limiter de la redondance.

- De créer une fonctionnalité où on peut activer ou désactiver certains panneaux selon le besoin.

II - Descriptive technique

=> VSCode : Visual Studio Code est un éditeur de code open-source. Pour pouvoir manipuler le code source, j'ai choisi cet IDE pour sa lisibilité fluide sur les codes et ses myriades de fonctionnalités spéciales.

=>Firefox : Firefox est un navigateur web libre et gratuit disponible pour les appareils.

Ce navigateur a été utilisé pour observer en temps réel l'affichage de l'application web, comprendre les problèmes « front end » lors du débogage.

=>Lighttpd : Lighttpd est un logiciel de serveur Web, rapide et flexible. Grâce au serveur, on peut faire afficher notre application web dans le navigateur web.

Terminal de linux : Le terminal de linux est un logiciel d'application, qui fournit une interface de ligne de commande (CLI) pour contrôler et effectuer des commandes.

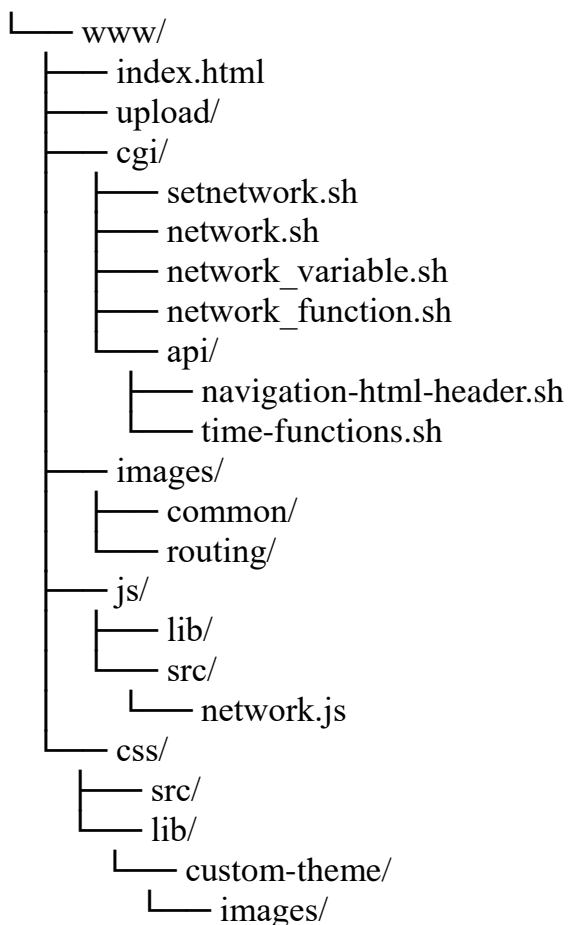
Le terminal a été utilisé pour déboguer les blocs de code, les fonctions, etc. dans les scripts qui sont exécutés dans la partie « back end ».

III - Maintenance application

Dans cette partie je vais réorganiser tous les dossiers/fichiers de manière à ce qu'ils soient structurés par leurs fonctions, par exemple, au lieu d'avoir les fichiers "network.sh" et "network.js" dans leurs dossiers respectifs (dossier js et dossier sh), nous les regrouperons dans un seul dossier appelé "Network" qui représentera la fonctionnalité network. L'idée et le but, qui correspondent à une des exigences, sont de regrouper et décortiquer les fichiers qui correspondent à une fonctionnalité dans leur dossier respectif, afin que dans l'avenir, selon le besoin, nous puissions attribuer différentes fonctions sur différents produits.

a) Réorganisation des répertoires et fichiers

Avant de commencer je vais illustrer une arborescent simplifier dans l'un des applications :



Comme on peut voir, dans les sous-répertoires « cgi » et « js » du « www », on retrouvera les fichiers qui s'occupe du l'affichage des page comme le fichier « network.sh », la modification de ces pages comme le fichier « setnetwork.sh » et enfin le fichier « network.js » qui agir comme un « lien » entre les deux fichiers mentionnés précédemment.

Le fichier « network_variable.sh » contient des variables propres a « network » qui va être appelé par les fichiers de « network », pareil pour « network_function.sh ».

C'est grâce à ces fichiers principale qu'on peut gérer la fonctionnalité Network.

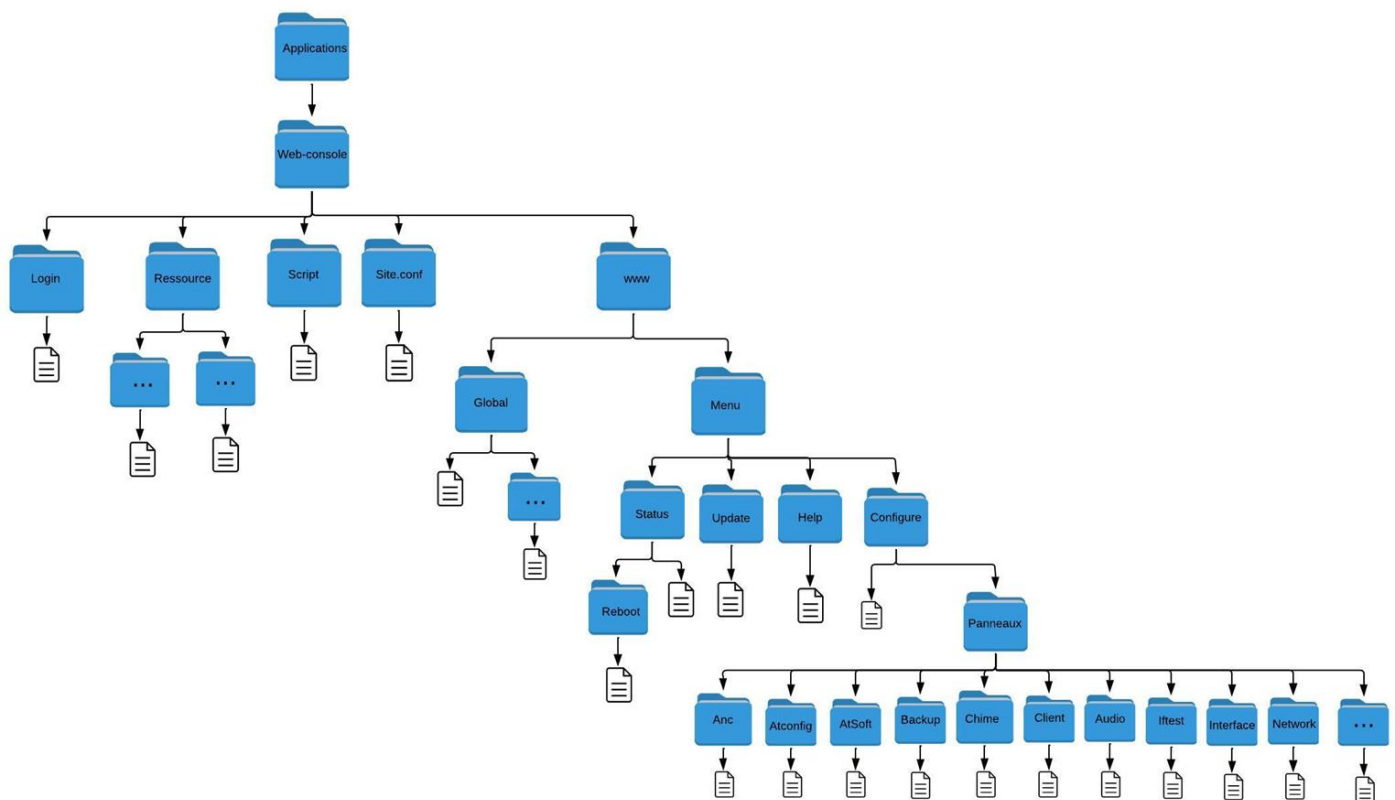
L'idée est de réorganiser ces fichiers dans une seul dossiers qui vas symboliser sa fonctionnalité, « Network », parce que, à la fin, on veut pouvoir gérer les fonctionnalités, c'est à dire, de leur désactiver ou pas selon les besoins.

Capture d'écran du fonctionnalité Network :



The screenshot displays the ATCONTR2 web interface for configuring network settings. The top header is blue with the 'Archean TECHNOLOGIES' logo on the left and the 'ATCONTR2' logo on the right. Navigation links for 'Status', 'Configure', 'Update', and 'Help' are located in the top right. Below the header is a tabbed menu with 'Network' selected. The 'Network settings' section shows the MAC address as 70:85:C2:E9:6A:22. Fields for Hostname (atcont-plateforme-dtu), Domain, IP address (10.175.40.122), Network mask (255.255.252.0), and Gateway (10.175.40.1) are present. Link settings include autonegotiation (on), speed (100 Mbps), and duplex (full). A 'Save' button is at the bottom left. The footer contains the copyright notice: © 2024, Archean Technologies - All rights reserved.

Pour cela j'ai propose comme réorganiser ce schéma ci-dessous :



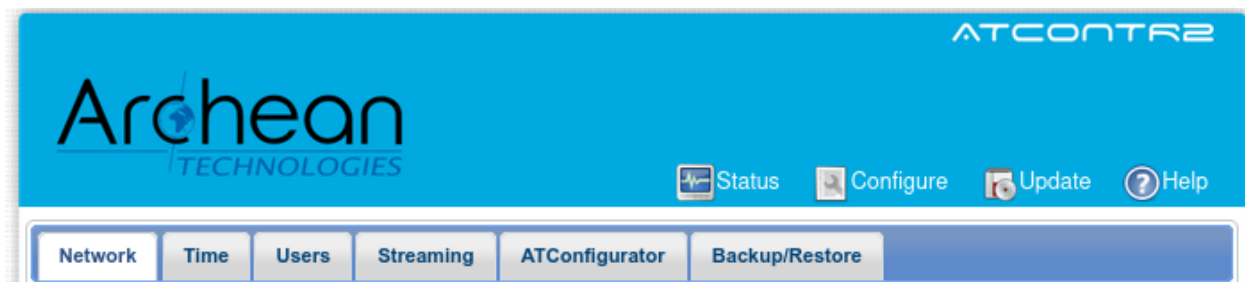
Comme on peut voir, à partir du sous-dossier « Menu », il est organisé de cette façon pour refléter la disposition de chaque page dans l'application web, par exemple, les quatre sous-répertoires de "Menu" ("Statut", "Configure", "Update", "Help"), représentent les quatre pages principales accessibles via la barre de menu.

Comme ci-dessous :



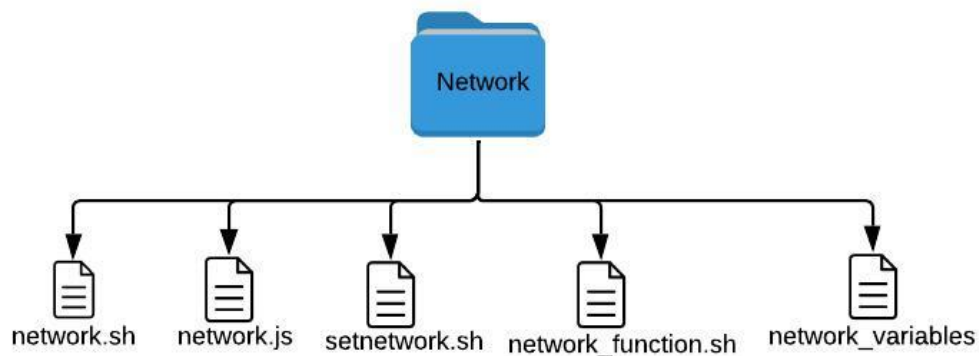
Le sous-dossiers « Configure » de « Menu » représente la page Configure, ceci est dit car la page Configure est une des page important dans l'application web, c'est grâce à cela on peut configurer le produit a travers leur fonctionnalité/panneaux.

Comme ci-dessous :



Grâce à cette organisation, on va retrouver les fichiers et répertoires qui correspondent à ces pages ou fonctionnalités/panneaux facilement dans un manière intuitive.

Démonstration détailler dans le sous-dossier « Network » dans « Configure »:



Dans schéma détailler, on peut voir que j'ai ajouté les fichiers qui correspondent à la fonctionnalité/panneau « Network » dans son répertoire respective. Chaque fichier joue un rôle distinct, par exemple, le fichier « network.sh » s'occupe de la partie front-end de la page « network », le fichier « setnetwork.sh » s'occupe de la partie back-end.

Dans certaines versions, il existe des fonctionnalités qui se situent dans une version et pas dans une autre, comme le panel « AtSoft » dans l'ancienne application « ATTOUCH-GH », « Streaming » dans « ATCONTR2 », « Client(s) » dans ATSSIP122 et etc.

ATTOUCH-GH :

ATTOUCH-GH

Archean

TECHNOLOGIES

Status

Configure

Help

Network

Time

Users

ATSoft

ATSoft settings

ATSoft hostname :

Save

© 2022, [Archean Technologies](#) - All rights reserved.

ATCONTR2 :

ATCONTR2

Archean

TECHNOLOGIES

Status

Configure

Update

Help

Network

Time

Users

Streaming

ATConfigurator

Backup/Restore

Network settings

MAC address : 70:85:C2:E9:6A:22

Hostname :

Domain :

IP address :

Network mask :

Gateway :

Link : autonegotiation : speed : duplex :

Save

© 2024, [Archean Technologies](#) - All rights reserved.

ATSIP122 :

Archean
TECHNOLOGIES

ATSIP122

Status

Configure

Update

Help

Network

Users

Client(s)

Backup/Restore

SIP - 1

SIP - 2

Proxy 1

Username : 52109

Password : 0000

IpAddress : 134.38.14.192

Proxy 2

Disabled proxy 1 ☒

Username :

Password :

IpAddress :

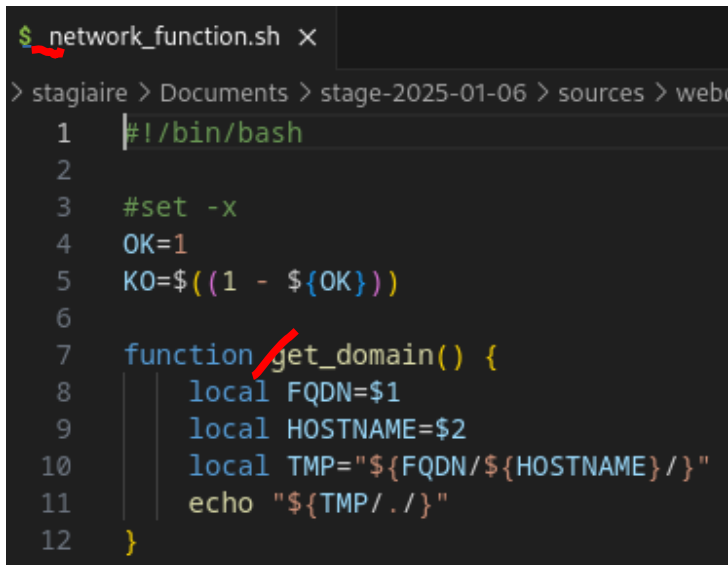
Save

Tous ces fonctionnalité sera ajouter dans la dernière version.

b)Nomenclature du code

1 – Context

Dans cet partie je vais contrôler la qualité du code, c'est à dire, vérifier si le sens de nommage des méthodes, variables, fichiers ou répertoire sont bien pertinent et distinguable pour bien répondre une des exigences.



```
$ network_function.sh x
> stagiaire > Documents > stage-2025-01-06 > sources > webv
1  #!/bin/bash
2
3  #set -x
4  OK=1
5  KO=$((1 - ${OK}))
6
7  function get_domain() {
8      local FQDN=$1
9      local HOSTNAME=$2
10     local TMP="${FQDN/${HOSTNAME}/}"
11     echo "${TMP}/."
12 }
```

Bien que le département informatique des entreprises puisse avoir ses propres conventions de nommage, il faudrait qu'elles soient appliquées et restent cohérentes dans l'ensemble du code source afin que certains blocs de codes, fonctions et méthodes puissent être distingués, ce qui fait partie des bonnes pratiques, ce n'est pas le cas dans cette capture d'écran et dans de nombreux autres fichiers.

Comme on peut voir, le nom du fichier « network_function.sh » et fonction « get_domain() » sont écrits en snake case, ce qui peut entraîner un non-respect des bonnes pratiques de codage parce que cela peut créer de la confusion.

Un exemple plus concret :

```
$ network_function.sh JS streaming.js X
ge-2025-01-06 > sources > webconsole-attouch-gh >
30
31 function computeStream() {
32     var format = $("#format").
33     var extra = $("#extra").ch
34
```

Comme on peut voir, le fonction « computeStream() » est écrit sous la forme camel case, ce qui nous montre le manque de cohérence.

Aussi les variables local qui sont en majuscule, ça peut entraîner de la confusion les variables environnemental car ils sont écrit en majuscule.

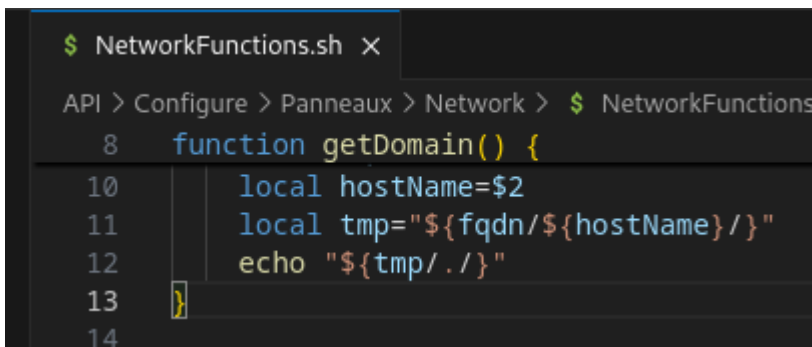
```
$ network_function.sh X
> stagiaire > Documents > stage-2025-01-06 > sources > webc
1  #!/bin/bash
2
3  #set -x
4  OK=1
5  KO=$((1 - ${OK}))
6
7  function get_domain() {
8      local FQDN=$1
9      local HOSTNAME=$2
10     local TMP="${FQDN}/${HOSTNAME}/"
11     echo "${TMP}/."
12 }
```

Exemple :

```
$ index.sh $ network_variable.sh X
Documents > stage-2025-01-06 > sources > webconsole-attouch-
4  #set -x
5
6  source "$(pwd)/network_function.sh
7
8  export NAMECARD=eth0
```

Pour répondre à cet problème, je vais utiliser la convention de java, qui est bien connue et souvent utiliser par les programmeurs et donc ça va rendre les codes et fichiers plus lisible et bien distinguable.

2 – Résultat



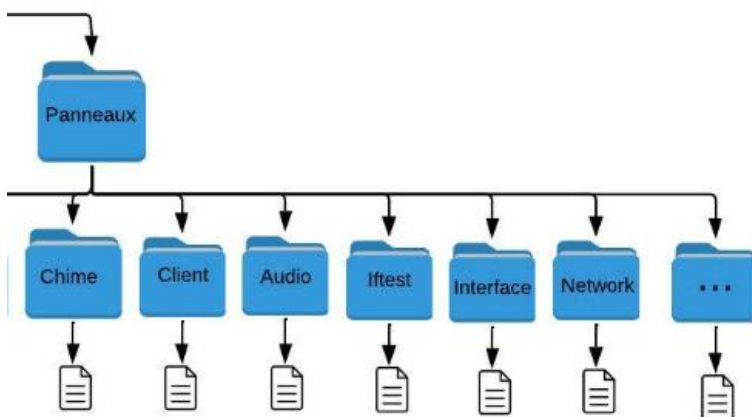
```
$ NetworkFunctions.sh x
API > Configure > Panneaux > Network > $ NetworkFunctions
8  function getDomain() {
10      local hostName=$2
11      local tmp="${fqdn/${hostName}/}"
12      echo "${tmp}/."
13  }
14
```

Comme on peut voir, les variables et fonctions sont transformées en écriture camel case et certains fichiers comme « NetworkFunctions.sh » sont écrits en pascal case. Cette règle de nommage peut faciliter la lisibilité et réduire la confusion dans le code source.

c) Re-factorisation du code source

L'une des principales exigences demandées par M. Hugo est de décortiquer chaque code source dans leurs propres fichiers respectifs, c'est-à-dire que si nous devons trouver un bloc de codes css dans exemple.sh, nous devrions séparer le code css de celui-ci et créer un nouveau fichier appelé exemple.css dans le même répertoire.

Rappel de l'architecture des répertoires et des fichiers :

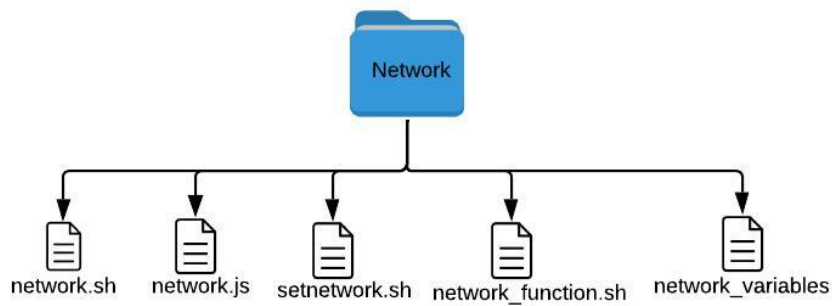


Les sous-répertoire de « Panneaux » symbolise

les panneaux/fonctionnalités dans la page « Configure » de l'application.

Le but toujours restent d'avoir un souple manipulation de ces fonctionnalité pour pouvoir gérer, c'est à dire, de les désactiver ou pas selon les besoins et aussi de ne pas avoir entraîné des codes appartenant au sous-dossier « Network » dans « Chime » en tant que sécurité, c'est à dire, si on active les panneaux « Network » et « Audio » dans un produit, on ne veut pas avoir les codes source des autres panneaux dans le produit.

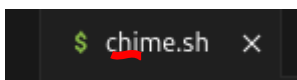
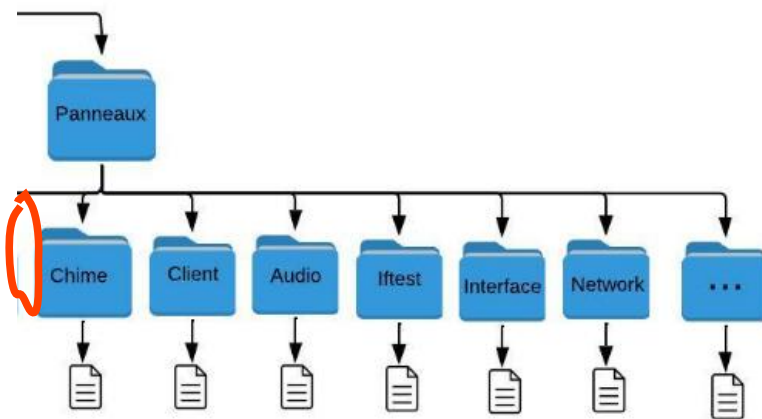
Revenons sur le schéma précédemment :



Les sous-dossiers doit ressembler comme cette schéma, où tous les fichiers sont compartimentés.

1 – Décortication du code

Je vais utiliser le fonctionnalité/panneau, « Chime », qui va représentent les autres changement aura pris dans les autres fonctionnalités/panneaux.



Le fichier « chime.sh » est principalement responsable d’afficher la page formulaire du chime, plus techniquement, il s’occupe de la partie front-end.

Capture d’écran de la fonction « displayInput » :

```
$ chime.sh x
home > stagiaire > Documents > s
56
57
58 displayInput() {
```

Capture d'écran du code JavaScript dans les balises HTML :

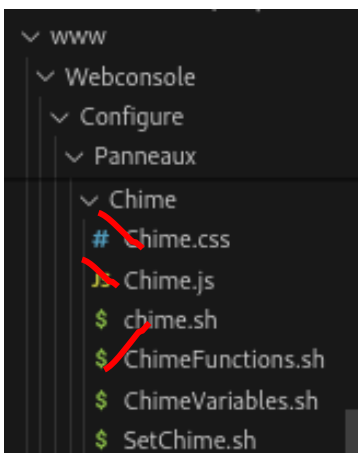
```
$ chime.sh x
home > stagiaire > Documents > stage-2025-01-06 >
117 cat <<-EOF
118 Content-type: text/html
119
120 <!DOCTYPE html>
121 <script type="text/javascript">
122 function verify_settings(){
```

Capture d'écran du code CSS dans les balise HTML :

```
$ chime.sh x
home > stagiaire > Documents > stage-2025-01-06 > s
296 </script>
297 <style type="text/css">
298 tr {
299 | height: 25px;
300 }
```

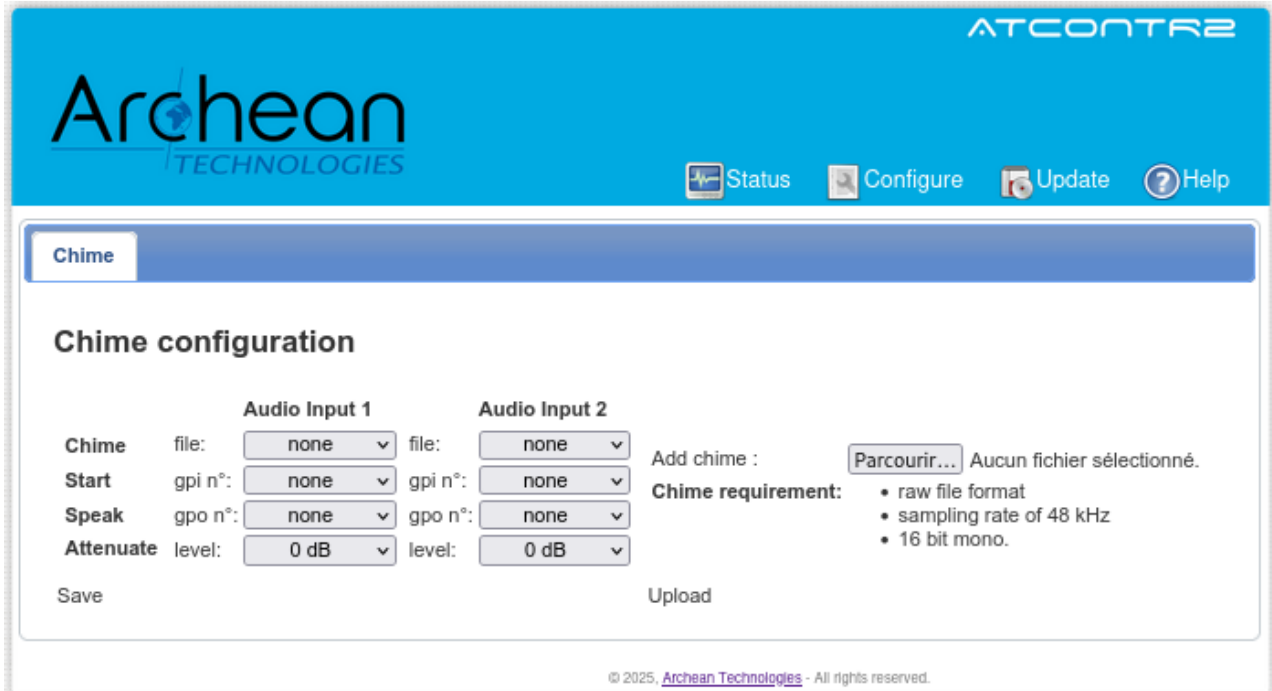
Comme on peut dans les captures d'écran, le fonction « displayInput() », des codes de css et javascript se trouve à l'intérieur du même fichier « chime.sh », ce que on ne voulait pas, juste pour nous le rappeler, l'une des exigences est de disséquer et de séparer différents types de code, comme le CSS et JavaScript, dans leur fichier respectif, la nécessité de faire cela est d'avoir une approche plus flexible et intuitive du code source et pour la maintenance dans l'avenir.

2 - Résultat de la séparation :



Cet image nous montre que les codes ont été bien séparés dans leur fichier respectif et ceci sera fait pour les autres fonctionnalités/panneaux.

3 - Test sur le navigateur web :



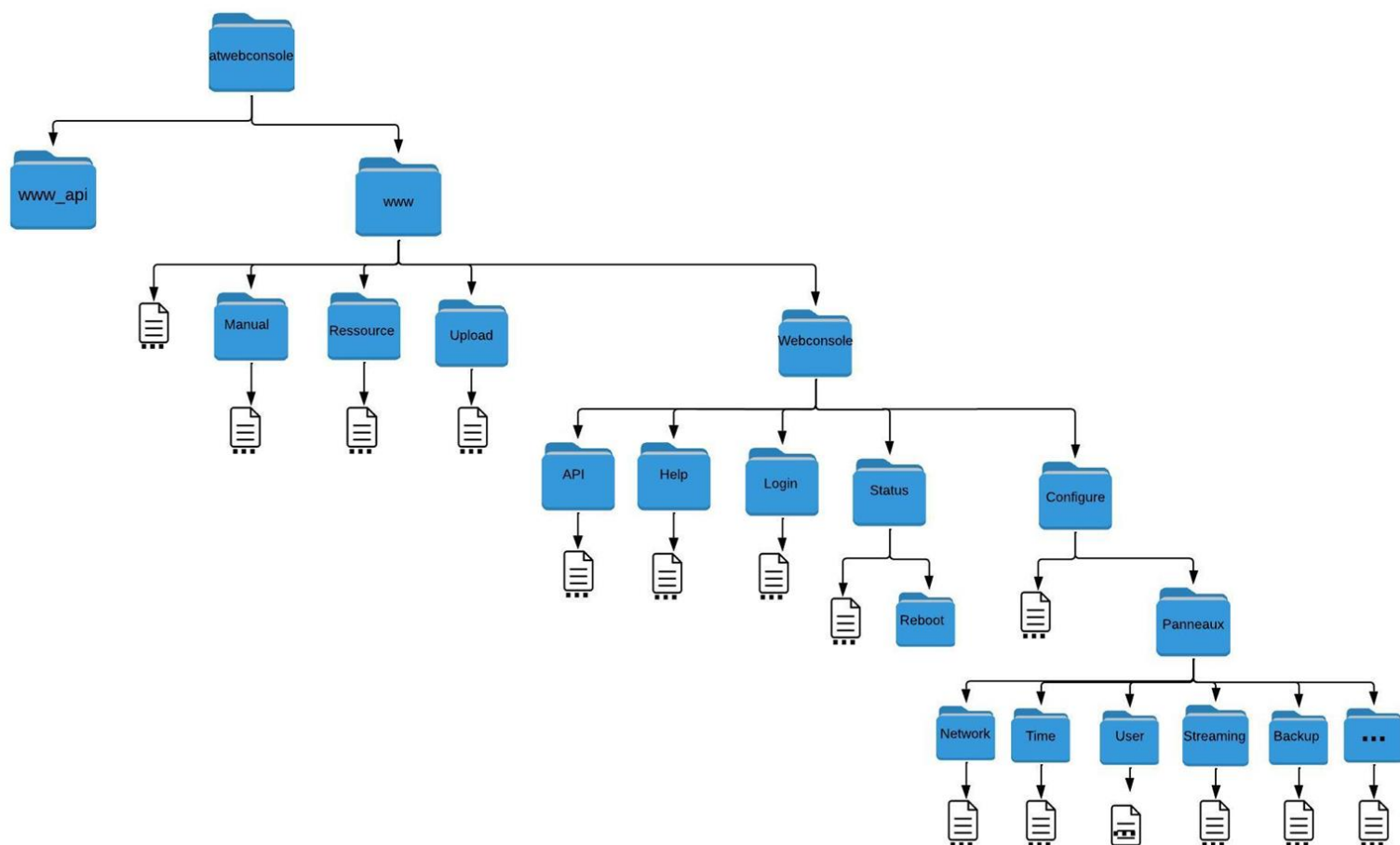
The screenshot displays the ATCONTROL2 web interface. The top navigation bar is blue with the 'Archean TECHNOLOGIES' logo on the left and 'ATCONTROL2' on the right. Navigation links for 'Status', 'Configure', 'Update', and 'Help' are present. The 'Chime' tab is selected, leading to the 'Chime configuration' section. This section includes two columns for 'Audio Input 1' and 'Audio Input 2', each with dropdown menus for 'file:', 'gpi n°:', 'gpo n°:', and 'level:'. To the right, there is an 'Add chime' button labeled 'Parcourir...' and a list of 'Chime requirement:' including 'raw file format', 'sampling rate of 48 kHz', and '16 bit mono.'. 'Save' and 'Upload' buttons are at the bottom. A footer note states '© 2025, Archean Technologies - All rights reserved.'

4 – Changement de l'architecture

Des modifications ont été apportées à l'architecture de l'application, concernant la structure des dossiers et des fichiers, pour des raisons de sécurité. Certains fichiers, comme ceux qui contiennent des variables et des fonctions qui « communiquent » avec le système, en dehors de l'application racine, risquaient d'être exposés à des personnes malveillantes, comme des pirates informatiques.

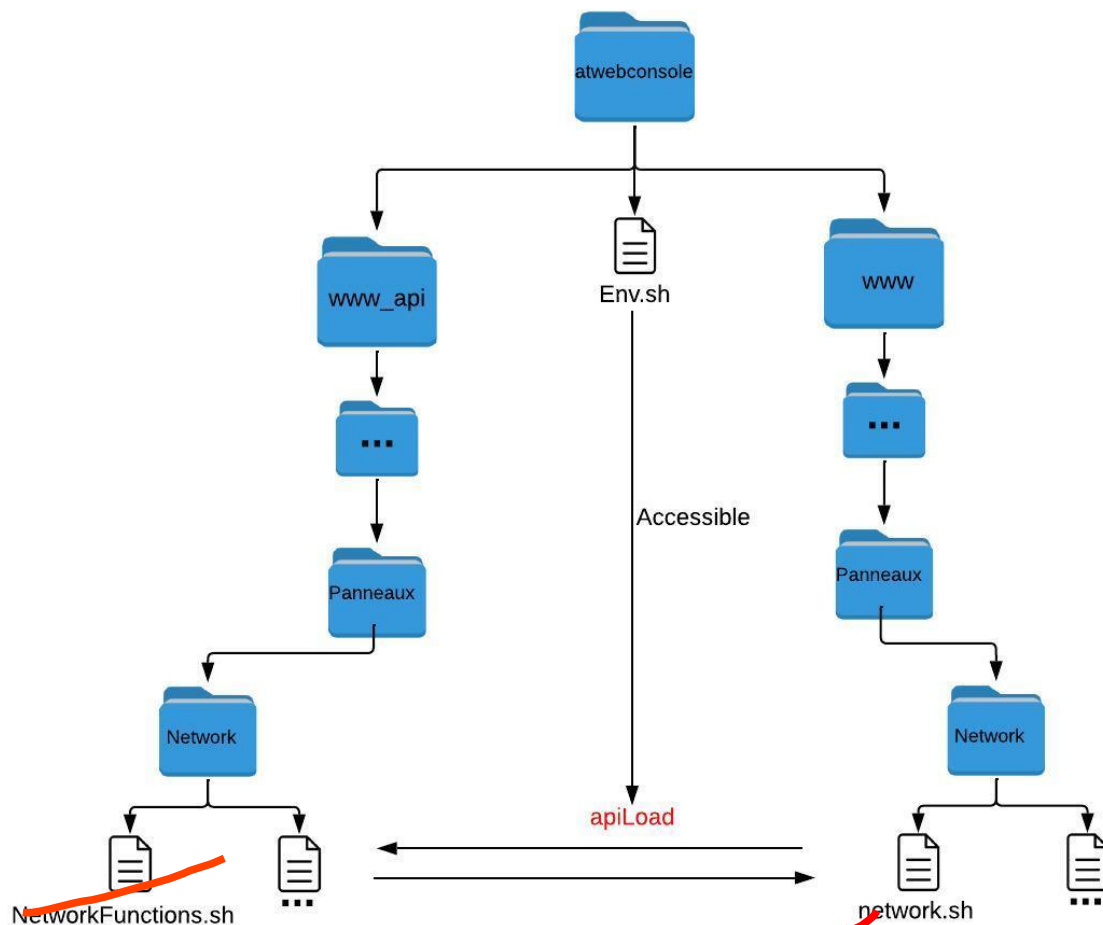
Nous avons donc eu l'idée de répliquer toute l'architecture, en partant de l'application racine "www", qui agirait comme un miroir de l'application racine d'origine et qui serait liée via une fonction qui calculerait le chemin du fichier demandé.

Nouvelle structure simplifier:



Comme nous pouvons le constater, la structure a évolué au fil du temps, aboutissant à la structure actuelle, comme nous l'avons dit précédemment, nous avons répliqué le répertoire racine de l'application Web dans "www_api", grâce à cela, les personnes malveillants ne pourront pas accéder aux fichiers sensibles en les saisissant via l'URL.

Démonstration de la « communication » entre les deux répertoire « www » à « www_api » :



Comme on peut le voir sur cette image, l'idée de répliquer le dossier "www" était pour deux raisons, pour des raisons de sécurité et pour faciliter la liaison entre "www_api" et "www" à travers une fonction appelée "apiLoad", qui calculerait le chemin en fonction du fichier source d'un autre fichier dans le même répertoire, comme on peut le voir avec "network.sh" et "NetworkFunctions.sh" qui sont situés dans le même répertoire. "network.sh" va sourcé "NetworkFunctions.sh" grâce à "apiLoad".

Problème rencontré :

Lors de cette décision ambitieuse, j'ai rencontré cette contrainte concernant la fonction "apiLoad" qui persistait, le problème était que pour faire fonctionner "apiLoad", j'aurais besoin de connaître le répertoire ou le chemin actuel du script source.

Par exemple, si "network.sh" source "NetworkFunctions.sh", cela fonctionnerait sans problème car ces deux fichiers sont situés dans le "même" nom de dossier mais si "network.sh" devait sourcer un fichier dans un répertoire complètement différent, cela ne fonctionnerait pas car il n'est pas situé dans le même nom de dossier.

j'ai trouvé un moyen de lutter contre ce problème mais cela nécessiterait que le système change de shell puisque l'application est maintenue par lui.

La solution était que si le shell s'exécutait sur bash, j'aurais utilisé "BASH_SOURCE".

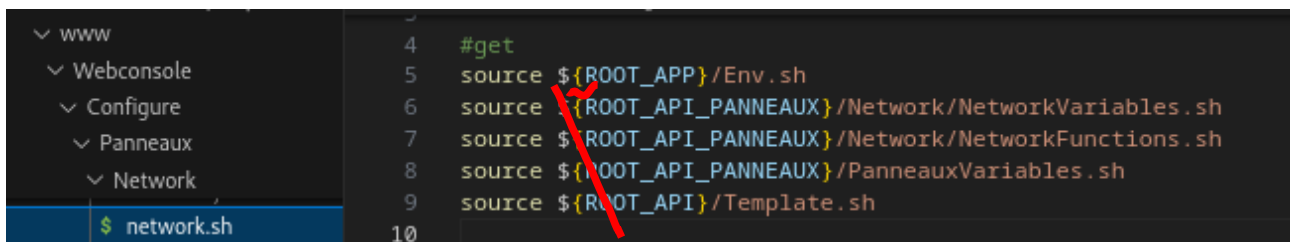
Parce que :

BASH_SOURCE est un tableau et contient les chemins d'accès aux fichiers sources des fonctions contenues dans \$FUNCNAME . \${BASH_SOURCE[i]} contient le fichier source, où la fonction \${FUNCNAME[i]} est définie.

Avec BASH_SOURCE savoir le chemin actuel d'un script même si il n'avait été exécuté, bien sûr il doit être sourcé par le script exécuter pour savoir son chemin.

Avec cette contrainte, j'ai dû trouver un autre moyen, j'ai gardé la même structure organisée au cas où Mr Hugo continuerait sur "apiLoad" ou trouverait une solution plus appropriée, j'ai donc profité de "Env.sh" et des variables menant à "www_api" et "www".

Capture d'écran du « liaison » entre « network.sh » et « NetworkFunctions.sh » :



```
4 #get
5 source ${ROOT_APP}/Env.sh
6 source ${ROOT_API_PANNEAUX}/Network/NetworkVariables.sh
7 source ${ROOT_API_PANNEAUX}/Network/NetworkFunctions.sh
8 source ${ROOT_API_PANNEAUX}/PanneauxVariables.sh
9 source ${ROOT_API}/Template.sh
10
```

Grâce au fichier "Env.sh", je pourrais accéder à ses variables comme "ROOT_API_PANNEAUX", comme son nom l'indique, il contient le chemin général du dossier "Panneaux" dans "www_api".

Capture d'écran de l'état actuel de l'application Web :

Page Status :

ATCONTR2

Archean

TECHNOLOGIES

 Status

 Configure

 Update

 Help

Status

Hostname : atcont-rd

Uptime

16:35:29 up 4 days, 39 min, load average: 1.30, 0.74, 0.59

Firmware

Running partition 2

Firmware version: 2.6v6.1 - codename: T2204-P1

Web version: 1.03.1

Restart

Reboot now!

© 2025, [Archean Technologies](#) - All rights reserved.

Page Configure :

ATCONTR2

Archean

TECHNOLOGIES

Status

Configure

Update

Help

NetworkTimeUsersClientInterfacesAudioSerialStreamingRoutingChime

ANCSupplyATSoftATConfiguratorVumeterBackup/Restore

Network settings

MAC address : 70:85:C2:E6:4D:6E

Hostname :

Domain :

IP address :

Network mask :

Gateway :

Link : autonegotiation : speed : duplex :

Save

© 2025, [Archean Technologies](#) - All rights reserved.

Page Update :

ATCONTR2

Archean

TECHNOLOGIES

Status

Configure

Update

Help

Update

Current version : 2.6v6.1 (codename: T2204-P1)

Select the firmware file below:

•

artiFirm.bin

Upload firmware: Aucun fichier sélectionné.

© 2025, [Archean Technologies](#) - All rights reserved.

IV – Gestion des panneaux

Malheureusement, je n'ai pas eu le temps de continuer cette partie.

Cette partie aurait consisté à la solution pour gérer les panneaux dans la page "Configurer", fait en sorte que grâce à un application externe on peut désactiver ou pas certain panneaux selon les produits.