BTS SN option IR
Systèmes Numériques option Informatique et Réseaux

E6.2 – PROJET INFORMATIQUE

Dossier de réalisation du projet.

Groupement académique : Besançon-Dijon	**Session : 2017**
Lycée ou Centre de formation : Lycée Général & Technologique Catherine & Raymond Janot	
Ville : Sens	
Nom du projet : Gestion d'un afficheur publicitaire	
Projet industriel : **OUI** **NON**	

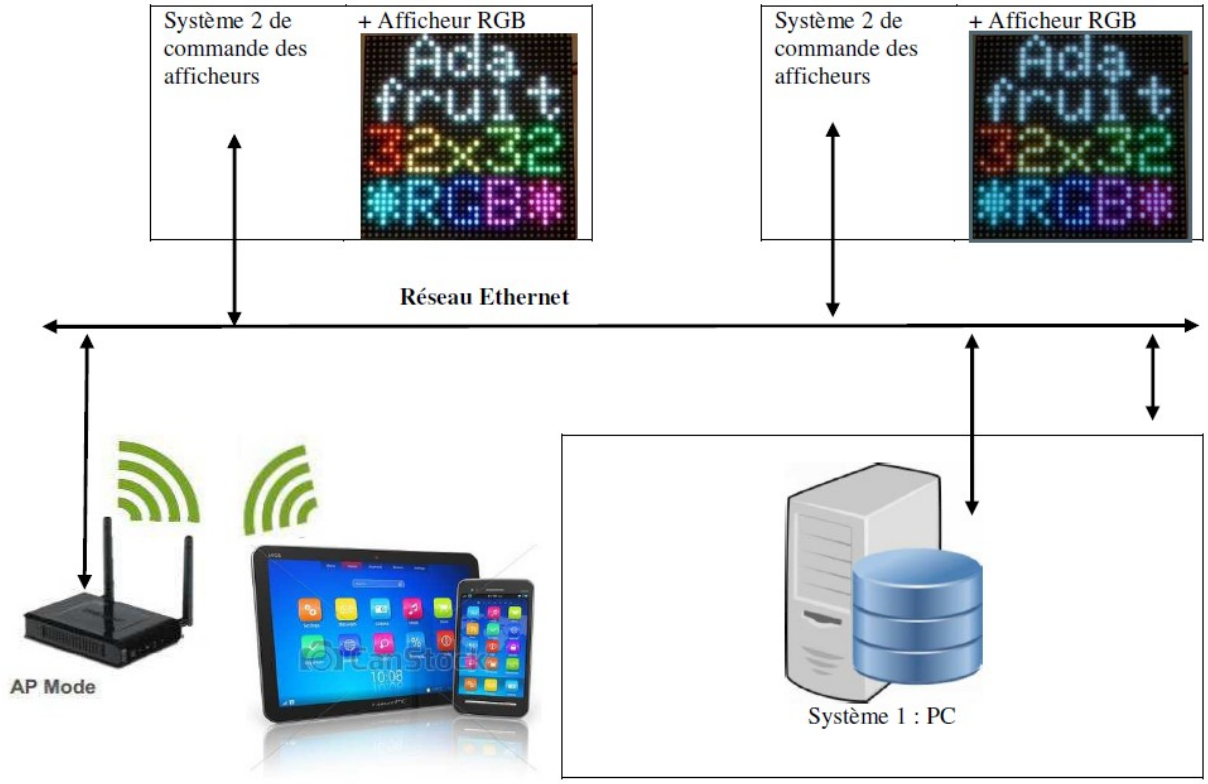
Équipe de développement.	
Professeur :	*Christophe Revy*
L'entreprise :	EBCONNECTIONS
Interlocuteur(s) de l'entreprise.	*M.BELOUET*
Etudiant 1 :	*Thiriau Julien*
Etudiant 2 :	*Grilly Jordan*
Etudiant 3 :	*Saez Gilles*
Etudiant 4 :	

Sommaire

1. Cahier des charges

1.1. Présentation du projet

L'entreprise demande de créer un système d'affichage autonome pour créer une animation visuelle avec les informations du jour en vitrine d'un magasin (messages et pictogrammes en couleur).



L'exemple donné est pour un système d'affichage autonome permettant une animation visuelle dans une pizzeria.
Il faudra pouvoir afficher les informations comme :

- Les heures d'ouverture et fermeture : horaires du déjeuner / horaire de diner ;
- Les promotions du jour (exemple : pizza calzone suivi du prix et d'un pictogramme correspondant) ;
- L'affichage (texte / pictogramme) pourra être fixe ou défilant, monochrome ou couleur ;
- L'affichage sera adapté automatiquement en fonction de l'heure de la journée ;
- Le responsable de la boutique programmera le contenu de l'affichage journalier à partir d'un ordinateur, d'une tablette ou d'un smartphone. L'affichage devra être réalisé en mode responsive ;
- Il faudra pouvoir sauvegarder cette programmation journalière (messages, pictogrammes, heure de début et de fin) afin de pouvoir le réutiliser ultérieurement ;
- L'interface de programmation devra être aussi simple et ergonomique que possible ;
- Le fonctionnement doit pouvoir être visualisé à distance via un ordinateur, une tablette ou un smartphone ;
- L'ensemble de la communication sera isolée dans un vlan et les liaisons wifi seront sécurisées ;

Il existe plusieurs configurations panneaux :

1.2 – Contraintes de réalisation

2.2.1/ Contraintes financières (budget alloué) : Le matériel étant fourni, pas de contrainte financière.
2.2.2/ Contraintes de développement (matériel et/ou logiciel imposé / technologies utilisées) :
Le matériel imposé :
- 2 x Panneaux de leds en matrice de 16 x 32 RGB ou autre format.
- ~~2 x ESP32~~ une Raspberry avec interface Ethernet ou autre carte.
- Commutateur (type 2960 Cisco) et point Wifi permettant la création de wlan (vlan sur Wifi).

Environnement de programmation imposé :
- Gestion des afficheurs en langage C ou C++.
- Gestion du serveur WEB en langage php en mode responsive.
- Librairie PIL (manipulation d'image en python) pour l'applicatif de transformation des fichiers couleurs

2.2.3/ Contraintes qualité (conformité, délais, …) :
Exigences qualité sur le produit à réaliser
Les logiciels devront être :
- Maniables, c'est-à-dire facile d'emploi pour l'opérateur. Les utilisations des sites Web devront être « intuitives » Des notices claires et précises devront assurer un emploi simple des logiciels.
- Robustes : programmes stables.
- Maintenables : en offrant une grande facilité de localisation et de correction des erreurs résiduelles.
- Adaptabilité : facilité de suppression, d'évolution de fonctionnalités existantes ou d'ajout de nouvelles fonctionnalités.
- Portabilité : minimisation des répercussions d'un changement d'environnement logiciel et/ou matériel.

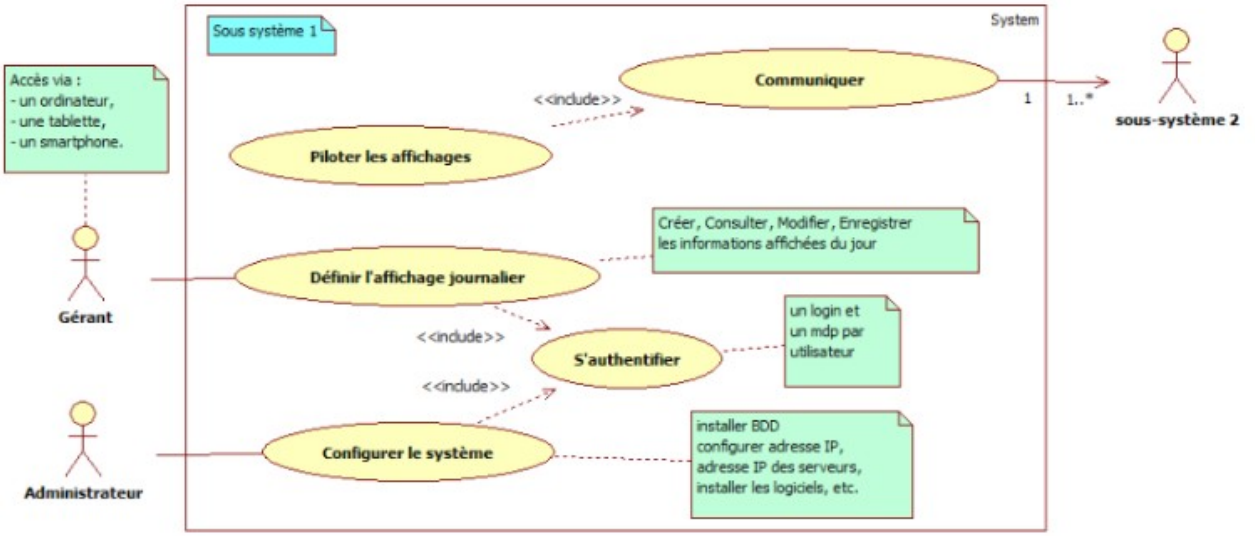
Tous les logiciels relatifs à l'application devront être livrables sur supports de stockage autonomes.
Les exigences qualité à respecter, relativement aux documents, sont :
- sur leur forme : respect de normes et de standards de représentation, maniabilité, homogénéité, lisibilité, maintenabilité,
- sur leur fond : complétude, cohérence, précision.

Les produits livrables du projet sont :
- la documentation,
- les codes sources et exécutables de l'application, ainsi que les fichiers de type projet s'ils existent.

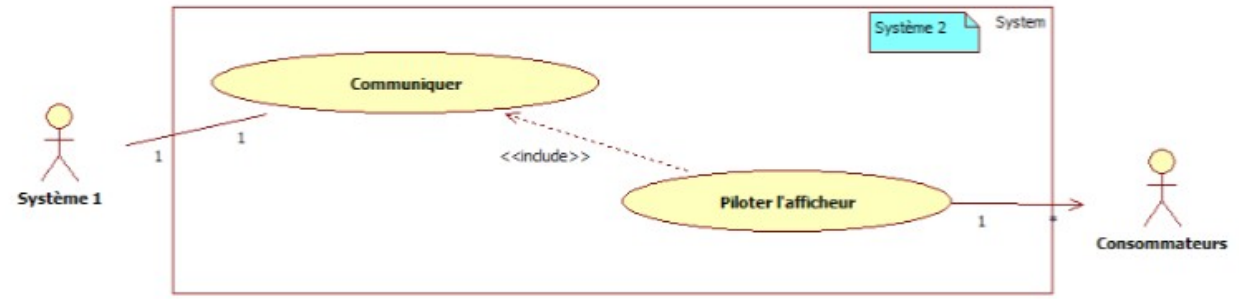
2. Analyse et spécification

2.1. Les cas d'utilisation

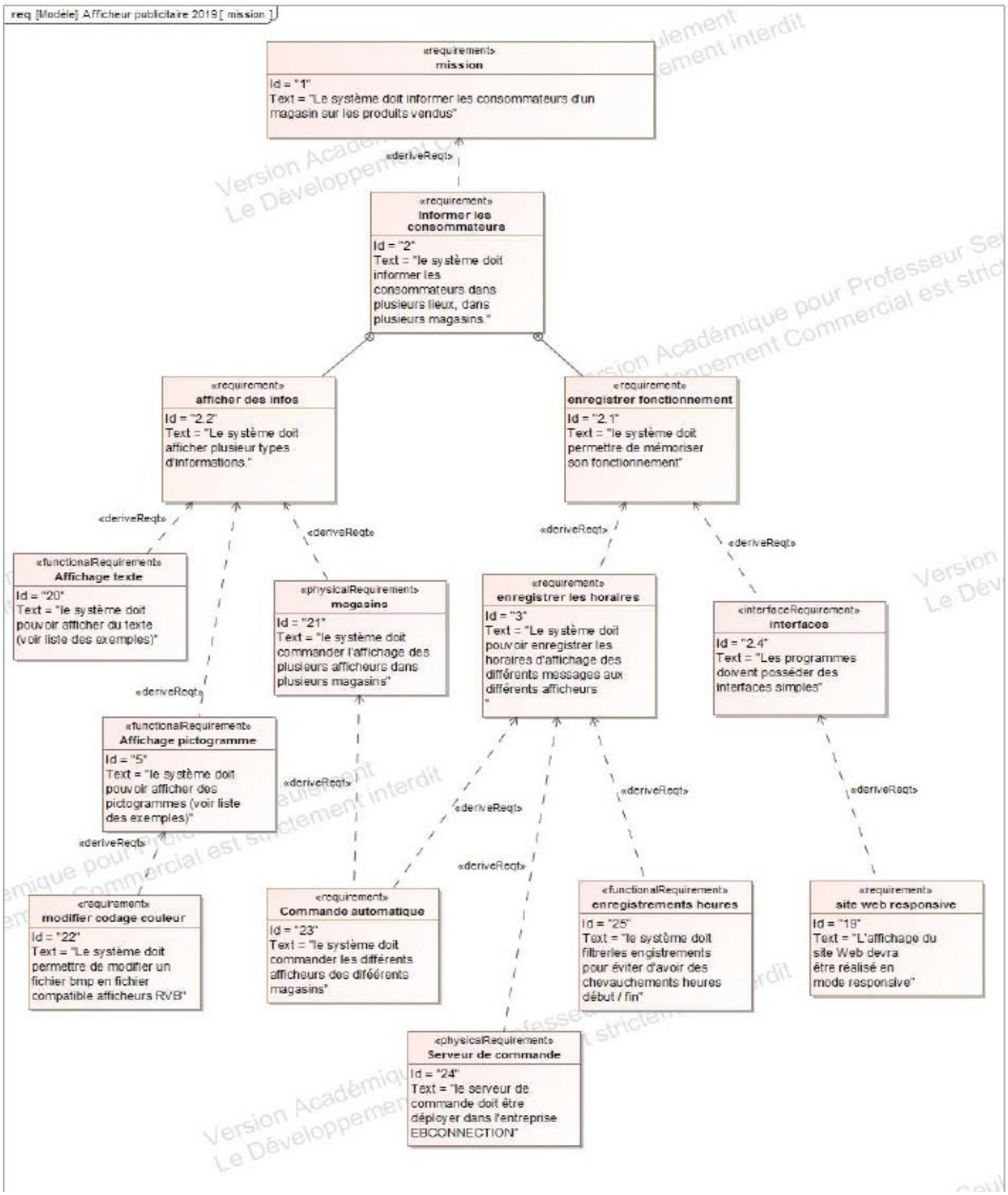
Les cas d'utilisation du sous-système 1 :



Les cas d'utilisation du système 2 :



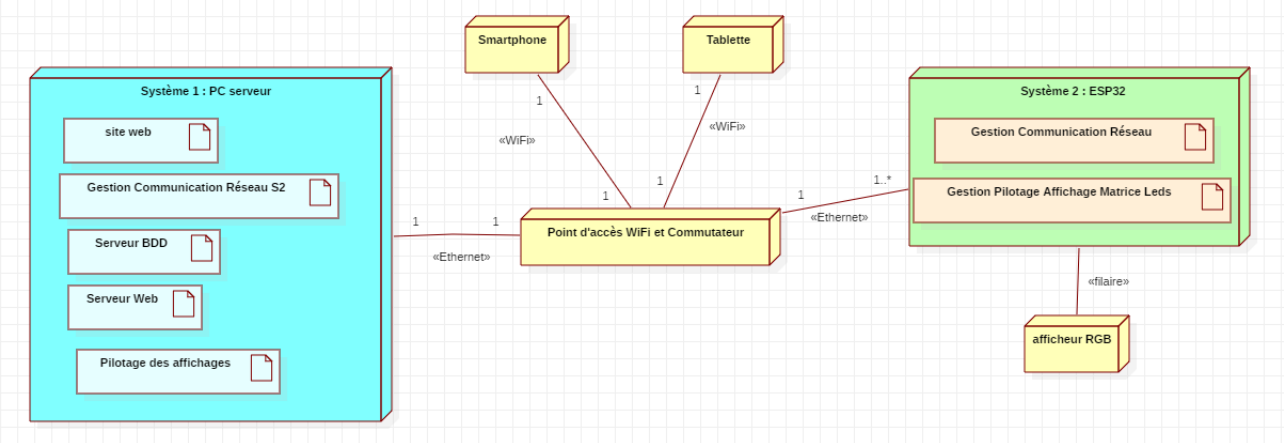
2.2. Les diagrammes d'exigences du système



2.3. Plan de test de validation

Action à vérifier	Tests à effectuer	Résultats attendus
Authentification utilisateurs	Entrer un nom d'utilisateurs ainsi qu'un mots de passe	L'utilisateur est connecté sur son compte
Définir l'affichage journalier	Entrer un planning	L'affichage est planifié et est bien envoyer
Commander afficheur	Configurer l'afficheur et afficher les images reçues	Afficheur configuré et affiche les images reçues
Sauvegarder planification	Vérifier la présence du planning dans la BDD	Planning existant dans la bdd
Communication entre PC et raspberry	Envoi de fichiers vers la Raspberry	Le fichier reçu sur la Raspberry
Piloter affichage	Afficher le contenu	Affichage présent sur les panneaux leds

3. Définition des échanges entre les sous-système



Un client à partir du site web de l'entreprise peut définir une planification d'images au format png, jpeg ou gif. Cette planification journalière sera ensuite envoyée par FTP à la raspberry qui traitera et enverra les images aux panneaux leds.

1 Présentation du travail du candidat N°1

La tâche qui m'a été confiée fût de réaliser l'interface web ainsi que de concevoir la base de donnée avec le candidat 2. De ce fait, j'ai visité le site de l'entreprise EBConnection afin de voir la charte graphique du site pour créer l'interface web dans le même ton.
On a ensuite réfléchi à la structure de la base de donnée afin qu'elle soit la plus optimisée possible. L'objectif de la base de donnée est de stocker les identifiants des utilisateurs (leur nom, leur mot de passe, le nom de leur société), stocker les images que l'utilisateur souhaite afficher selon un planning prédéfini. Un planning est un ensemble d'images avec, pour chacune, une heure de début. C'est ce planning qui sera envoyé sur l'afficheur. Ce dernier exécutera ensuite les images dans l'ordre en suivant leurs horaires. La base de donnée stocke aussi la mise en forme des panneaux de l'utilisateur, c'est-à-dire leur disposition, que l'on appellera par la suite *afficheur* dans le code et la base de données.

1.1 Prototypage et maquettage

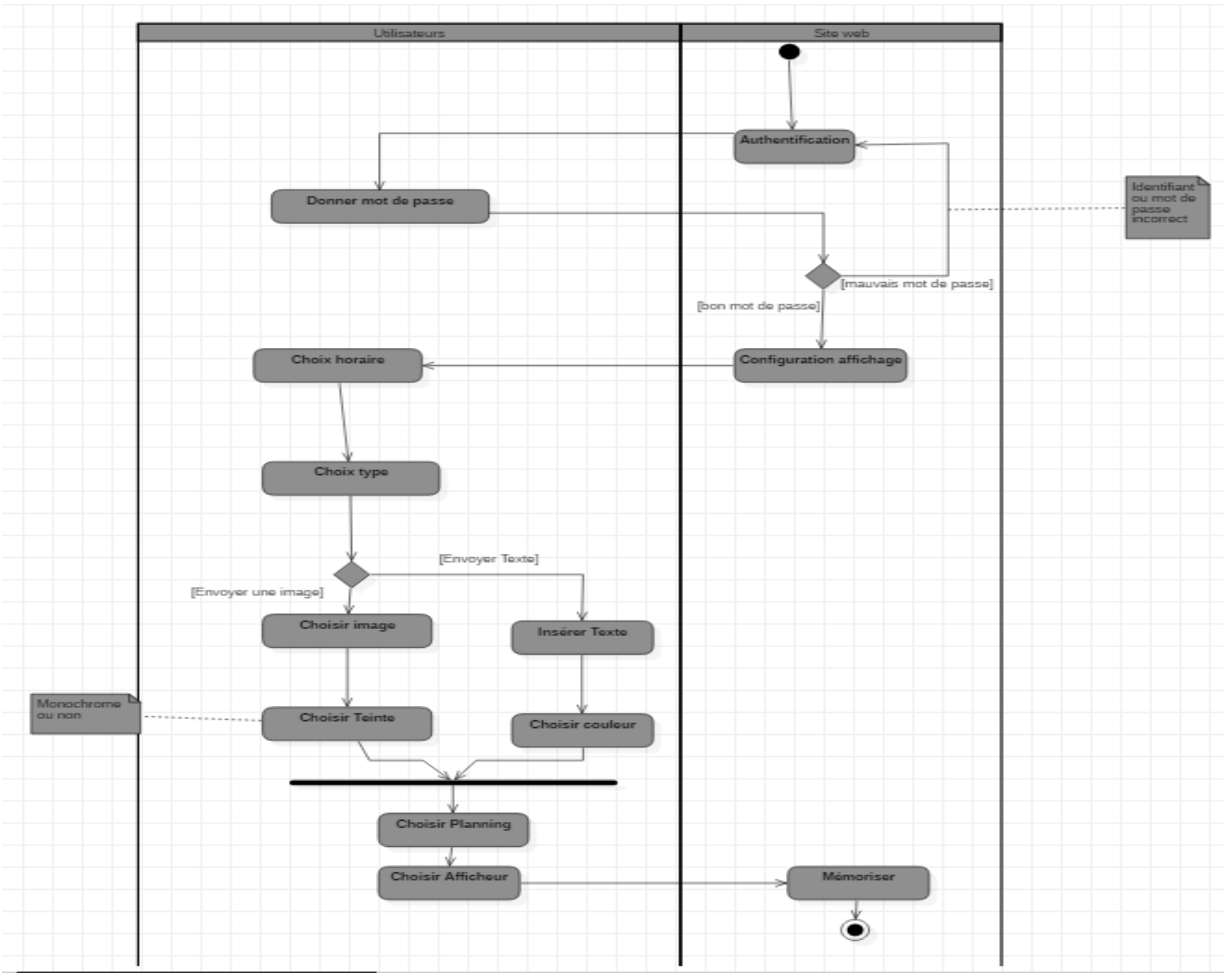
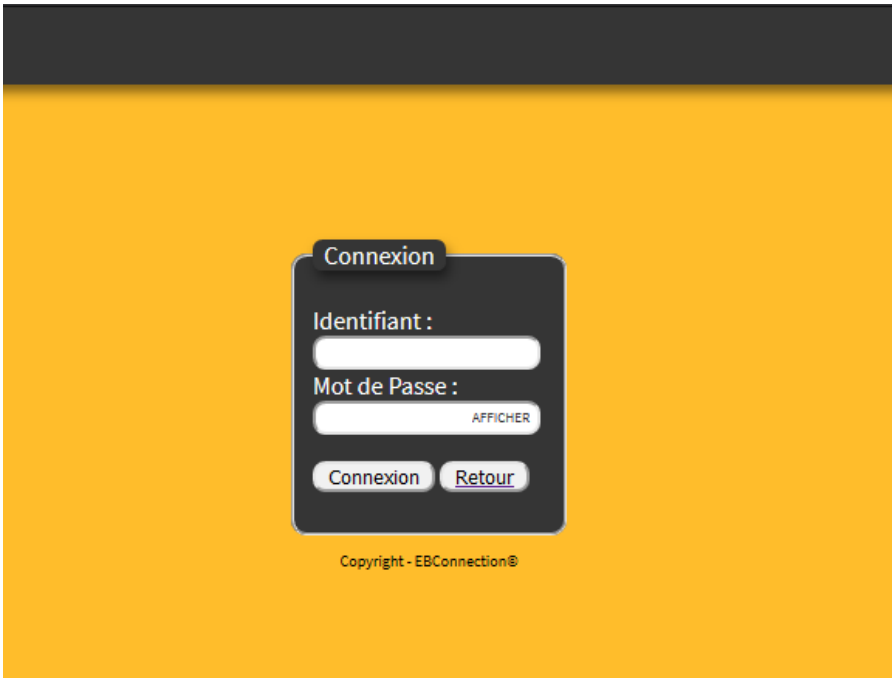


Diagramme d'activité

Le diagramme d'activité précédent présente l'utilisation principale du site web, c'est-à-dire l'envoi d'une image ou d'un texte tout en passant par la phase d'authentification nécessaire afin de différencier un utilisateur d'un autre et de ce fait lui présenter les bonnes informations.

Présentation des IHM du site Web

L'authentification se fait sur la page « Index.php », l'IHM se présente sous cette forme :



Si un champ est manquant ou incorrect, un message apparaît en rouge spécifiant l'origine de l'erreur.

Champs manquants *Champs incorrects*

Si les informations sont valides, l'utilisateur accède à la page d'accueil, sauf lors de la première connexion ou si l'utilisateur n'a pas défini d'afficheur. Dans ce cas, l'utilisateur est redirigé sur la page « afficheurs/afficheur.php » afin qu'il puisse en définir un. Il ne peut accéder à la page d'accueil que si un afficheur est défini.

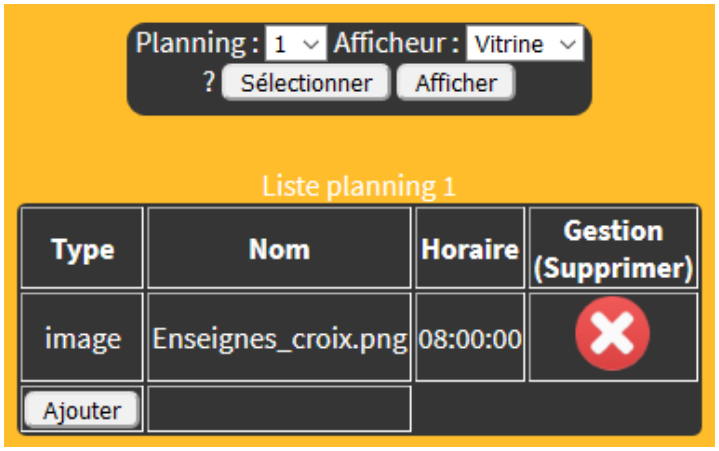
Un afficheur est un ensemble de panneaux LED physique. L'afficheur sert à spécifier la disposition des panneaux LED afin que l'image envoyé s'affiche normalement.

Pour définir l'afficheur, il faut cliquer sur « Ajouter », la page « afficheurs/nouvel_afficheur.php » s'affiche pour renseigner les différents paramètres de l'afficheur : Le numéro de l'afficheur (si c'est le premier afficheur, la valeur sera 1), le nom de l'afficheur (comme l'emplacement des panneaux), la hauteur donc le nombre de panneaux en hauteur puis la même chose en longueur, et enfin l'adresse IP. L'adresse IP de la Raspberry sur laquelle sont branchés les panneaux LED qui sera donnée par l'installateur.

Nouvel afficheur

Une fois les paramètres enregistrés, l'utilisateur est redirigé sur « afficheurs/afficheur.php » où il verra son afficheur enregistré. Il peut bien sûr le supprimer, mais comme il n'aura plus d'afficheur il sera redirigé sur cette même page pour qu'il puisse le définir.

L'afficheur défini, l'utilisateur a accès à la page d'accueil, « user/interface.php ». Cette page permet à l'utilisateur de visualiser le contenu des plannings - donc les images qu'il a enregistrées, les supprimer, en ajouter, et d'envoyer un planning sur l'afficheur.



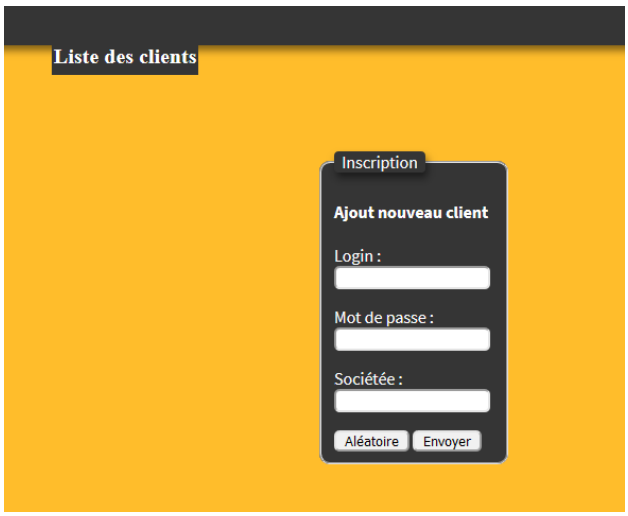
Page d'accueil

Pour envoyer un planning, l'utilisateur sélectionne le planning en question, puis l'afficheur, et clique sur « Afficher » qui activera la lecture le code XML de l'étudiant 2.
L'interface Web possède aussi une IHM permettant de changer le mot de passe et de voir les informations du compte.

Compte

Administrateur

Un compte administrateur possède un onglet, uniquement visible par lui, appelé « Administrateur ». Cet onglet permet la création d'un nouvel utilisateur et de visualiser la liste de tous les utilisateurs de la solution.



Liste des clients

Ci-dessus se trouve l'IHM d'ajout d'un nouveau client, avec son login, son mot de passe et le nom de sa société. Un onglet « Liste des clients » est présent au-dessus et permet d'accéder à l'IHM présentant la liste des clients.

1.2 Base de données

Notre système requiert le stockage de plusieurs données notamment en lien avec le site web, comme les identifiants de connexion des utilisateurs mais aussi leurs images et leurs afficheurs. De ce fait nous avons créé une table « user », une table « image », une table « afficheurs » ainsi qu'une table « etapes » qui est l'équivalent des plannings, c'est-à-dire que cette table contient des informations supplémentaires comme l'heure des images ainsi que leurs plannings.

1.3 Authentification

1.3.1 Code d'authentification

Afin de gérer la connexion des utilisateurs, l'accès à la page d'accueil se fait grâce à un formulaire qui va récolter les informations et vérifier si elles sont présentes dans la base de données, plus précisément dans la table « user ». Nous aurons besoin de plusieurs informations concernant l'utilisateur connecté lors de la navigation sur le site web. Pour cela nous utiliserons les variables dites *global* pour garder en mémoire le login du client, son identifiant et son privilège (administrateur ou non). Nous démarrons donc une session avec la fonction *session_start().*

Formulaire (html)

```
  <form method="post" action="#">
          <fieldset id="connexion">
                <legend>Connexion</legend>
                      <p><label for="pseudo">Identifiant :</label><input
name="login" type="text" title="Saisir votre login" id="pseudo" /><br />
                         <label for="password">Mot de Passe :</label><input
type="password" name="pass" title="Saisir votre mot de passe" id="password"
maxlength="10"/>
                         <span class="show-password">afficher</span></p>
                      <p><input type="submit" value="Connexion"
name="Connexion" title="Se connecter" />
                          <a href="http://www.ebconnections.com/"><input
type="button"  value="Retour" title="Site EBConnection"/></a></p>
```

```
          </fieldset>
      </form>
Code de vérification du formulaire (php)
<?php
  session_start (); //On démarre une nouvelle session

if(isset($_POST['Connexion'])) //vérification de l'envoi du formulaire
{
  if(!empty($_POST['login']) && !empty($_POST['pass'])) //Si le login et le
mot de passe sont remplit
  {
   // connexion à la base de donnée via l'objet PDO
    $bdd = new
PDO('mysql:host=localhost;dbname=afficheur_publicitaire;charset=UTF8',
'jordan', 'afficheur2019');
// requête SQL pour identifier
        $req = $bdd -> prepare ('SELECT * FROM user WHERE login = ? and mdp =
? ');
        $req -> execute(array($_POST['login'],$_POST['pass']));

        //verification des identifiant
        if($data = $req -> fetch())
        {
          //On passe en variable de session le login et l'identifiant de
l'utilisateur ainsi que le privilège (administrateur ou non) afin de les
réutiliser sur d'autres pages
          $_SESSION['login'] = $data['login'];
          $_SESSION['idUser'] = $data['idUser'];
          $_SESSION['privilege'] = $data['privilege'];
          $req->closeCursor();
          header ("Location: user/interface.php"); // Redirection sur la page
d'accueil
          exit;
        }
        else
        {
 //Erreur d'identifiants
          echo "<div class='alert'>Verifiez vos informations de
connexion</div>";
        }
  }
  else {
 //Champs manquants
    echo "<div class='alert'>Veuillez-renseignez vos informations de
connexion</div>";
    }
}
else {
}?>
```

1.3.2 Test unitaire de l'authentification

Action à réaliser	***Tests à réaliser***	***Résultats obtenus***
Connecter l'utilisateur	Remplir le formulaire avec des informations valides	L'utilisateur est connecté
Connecter l'utilisateur avec des informations incorrectes	Remplir le formulaire avec des informations invalides	La page indique que les identifiants sont faux
Connecter l'utilisateur avec un champ vide	Remplir qu'un seul champ du formulaire	La page indique que l'un des champs est manquant

1.4 Page d'accueil

1.4.1 Vérifications de la page d'accueil

L'accès à la page d'accueil se fait donc via une connexion valide de l'utilisateur mais la page en elle-même présente d'autres vérifications. Tout d'abord la vérification de la variable *global* du login de l'utilisateur afin d'être sûr que l'utilisateur n'a pas eu accès à la page d'accueil en passant par l'URL.

```
//Sécurité : redirection si la variable est vide
if(empty($_SESSION['login']))
{
  header('Location: ../index.php'); //Redirection sur la page de connexion
}
```

Ensuite, on vérifie la présence de l'afficheur, afin de rediriger ou non l'utilisateur sur la page d'ajout d'afficheur. Pour cela il faut effectuer une nouvelle requête SQL permettant de récupérer, et, s'ils existent, les identifiants des afficheurs de l'utilisateur connecté et donc une nouvelle connexion à la base de données :

```
<?php
$bdd_verif_afficheur = new
PDO('mysql:host=localhost;dbname=afficheur_publicitaire;charset=UTF8',
'jordan', 'afficheur2019');

//Requête SQL afin de sélectionner les identifiants utilisateurs  pour
l'utilisateur connecté
$verif_afficheur_bdd = $bdd_verif_afficheur -> prepare('SELECT idAfficheurs
FROM afficheurs WHERE User_idUser = ?');
$verif_afficheur_bdd -> execute(array($_SESSION['idUser']));

$verif_afficheur = $verif_afficheur_bdd ->fetch();

if($verif_afficheur['idAfficheurs']=='') //Si il n'y a aucun afficheur pour
cet utilisateur on le redirige sur la page afficheur
{
    header('Location: ../afficheurs/afficheur.php');
}
?>
```

1.4.2 Tests unitaires des vérifications de la page d'accueil

Actions à réaliser	***Tests à réaliser***	***Résultats obtenus***
Accéder à la page d'accueil sans connexion	Saisir l'URL de la page dans le navigateur	Redirection sur la page de connexion
Aucun afficheur défini	Première connexion	Redirection sur la page d'ajout d'afficheur
Un ou plusieurs afficheur(s) défini(s)	Compte possédant déjà des afficheurs	Accès à la page d'accueil

1.4.3 Onglet administrateur

L'onglet administrateur ne doit être visible uniquement par les utilisateurs possédant ce titre. L'administrateur est représenté dans la base de données par la colonne « privilege ». Pour tous les utilisateurs, cette colonne est nulle sauf pour l'administrateur qui a comme valeur 1. Grâce à ce paramètre, on peut définir la visibilité de l'onglet avec le code suivant :

```
<?php
//Si la variable de session du privilège vaut 1 alors l'utilisateur est un
administrateur
if($_SESSION['privilege'] == 1)
{
 //Afficher l'onglet
 echo '<li><a class="mm" href="admin.php"> Administrateur </a></li>';
}
?>
```

De cette manière, seul l'administrateur aura accès à cet onglet.

1.4.4 Envoi de planning

La page d'accueil, soit « user/interface.php », inclut une autre page, « afficheurs/récapitulatif.php » qui se charge d'afficher le tableau contenant les images en fonction du planning de l'utilisateur, avec comme option le choix de supprimer ou d'ajouter des images. Pour cela, et dans le but d'envoyer des images sur le bon afficheur, on récupère le nom de(s) l'afficheur(s) de la page « afficheurs/afficheur.php » :

```
$req =$bdd->prepare('SELECT nom_afficheur FROM afficheurs, user where
user.idUser = afficheurs.User_idUser and User_idUser = ?');
$req -> execute(array($_SESSION['idUser']));
```

Puis on l'affiche dans une *listbox* comme pour le planning :

```
 <form id='form_Recap'  action="#" method="post">
 Planning :
    <SELECT name="Planning_choice" size="1">
        <OPTION>1</option>
          <OPTION>2</option>
            <OPTION>3</option>
              <OPTION>4</option>
                <OPTION>5</option>
                  <OPTION>6</option>
                    <OPTION>7</option>
      </SELECT>
    <!-- Sélection afficheur -->
    Afficheur :
    <SELECT name="Afficheur_choice" size="1">
      <?php
      while ($data = $req -> fetch()) {
        echo '<option>'.$data['nom_afficheur'].'</option>'; //Affichage de
tous les afficheurs
      }
      ?>
    </SELECT>
 <input type="submit" name="Select_planning" value="Sélectionner"/>
<input type="button" name="Gen_xml" value="Afficher"/>
</form>
```

Quand l'utilisateur clique sur le bouton « Afficher », cela exécutera le code XML du candidat 2 tandis que le bouton « Sélectionner » affichera le contenu du planning choisi dans la listbox « Planning_choice ».

1.4.5 Affichage du contenu d'un planning

```
<!-- Tableau afficher contenu planning -->
<table border="1" id="table_recap">
  <caption>Liste planning <?php echo $_POST['Planning_choice']; ?></caption>
  <tr>
  <th>Type</th> <!-- Ligne--!>
  <th>Nom</th>
  <th>Horaire</th>
  <th>Gestion (Supprimer)</th>
  </tr>

<?php
//connexion bdd
$bdd = new
PDO('mysql:host=localhost;dbname=afficheur_publicitaire;charset=UTF8',
'jordan', 'afficheur2019');

//Requête SQL afin d'afficher les images du planning choisit et de
l'utilisateur
$Afficheur_planning = $bdd -> prepare('SELECT * from afficheurs,
afficheurs_has_etapes, etapes, image, user where afficheurs.idAfficheurs =
afficheurs_has_etapes.Afficheurs_idAfficheurs and afficheurs.User_idUser =
user.idUser and afficheurs_has_etapes.etapes_idEtapes = etapes.idEtapes and
image.idimage = etapes.image_idimage and idUser = ? and planning = ?');
```

```
$Afficheur_planning -> execute(array($_SESSION['idUser'],
$_POST['Planning_choice']));

while ($data = $Afficheur_planning -> fetch()) {
// on affiche les images (denomination= image/texte, fichier=nom, heure)
        echo '<tr><td>'.$data['denomination'];
 echo '<td>'.$data['fichier'];
        echo '<td>'.$data['heure'];

//Bouton supprimer image (ref. 1.4.6)
        echo '<td><a  href="../user/delete.php?id='.$data['idimage'].'"><img
id="del" src="../css/del.png" alt="del"></a></tr>';
}
 ?>
//Bouton pour ajouter une image
<td><input type="button" value="Ajouter" onclick="Ajouter()"></td><td></td>

</table>
```

1.4.6 Bouton supprimer

Le bouton « supprimer image », qui est une icône, permet de supprimer l'image voulue. Pour cela, on passe par la méthode GET (c'est-à-dire par l'URL) l'identifiant de l'image souhaité. On l'envoie ainsi sur la page « user/delete.php », ce qui donne, dans l'URL : « ../user/delete.php?id=(identifiant de l'image) ». On supprime donc l'image par son identifiant. Le code de la page « user/delete.php » est donc :

```
//////////////image//////////////////////////
$idImg = $_GET['id'];
$bdd = new PDO('mysql:host=localhost; dbname=afficheur_publicitaire',
'jordan', 'afficheur2019');
if(isset($_GET['id']))
{
//Supprime l'image dans la table image en fonction de son id
$supprime_img= $bdd -> prepare('DELETE FROM image WHERE idimage=?');
$supprime_img -> execute(array($idImg));

//Supprime l'image dans la table etapes en fonction de son id
$supprime_etapes = $bdd -> prepare('DELETE FROM etapes WHERE
image_idimage=?');
$supprime_etapes -> execute(array($idImg));

//Supprime l'image dans la table Afficheurs_has_etapes
$supprime_afficheur = $bdd -> prepare('DELETE FROM afficheurs_has_etapes
WHERE etapes_idEtapes=?');
$supprime_afficheur -> execute(array($idImg));
}
```

La page « user/delete.php » permet aussi de supprimer les afficheurs de l'utilisateur. Afin de pouvoir le différencier de la suppression de l'image, l'URL pour rejoindre la page « /user/delete.php » lors de la suppression d'un afficheur est : /user/delete.php? idAff=(identifiant de l'afficheur).

```
$idImg = $_GET['id'];

///////////////Afficheurs////////////////
if(isset($_GET['idAff'])) //Si on récupère bien une valeur dans l'URL
{
//Supprime l'afficheurs dans la table afficheurs
$supprime_afficheur = $bdd -> prepare('DELETE FROM afficheurs WHERE
idAfficheurs=?');
$supprime_afficheur -> execute(array($idAff));

//Supprime l'afficheurs dans la table afficheurs_has_etapes
$supprime_afficheur = $bdd -> prepare('DELETE FROM afficheurs_has_etapes
WHERE Afficheurs_idAfficheurs=?');
$supprime_afficheur -> execute(array($idAff));}
```

1.4.7 Déconnexion d'un utilisateur

```
<form  action='#' method="post">
        <i>Changer votre mot de passe : <input type="password"
name="changer_mdp" />
      <p>  Entrez le de nouveau : <input type="password"
name="confirm_mdp"/><p></i>
        <input type="submit" name="Valider_mdp" value="Valider" />
        </form>
```

Le traitement

```
//Si champs remplit
if(!empty($_POST['changer_mdp']) && !empty($_POST['confirm_mdp']))
{
  if ($_POST['changer_mdp'] == $_POST['confirm_mdp']) //Si les deux champs
sont identiques
  {
    //connexion bdd et requête SQL pour mettre à jour la bdd
    //avec le nouveau mot de passe de l'utilisateur
    $bdd = new
PDO('mysql:host=localhost;dbname=afficheur_publicitaire;charset=UTF8',
'jordan', 'afficheur2019');
    $new_mdp = $bdd -> prepare('UPDATE user SET mdp = ? WHERE user.idUser
= ?');
    $new_mdp -> execute(array($_POST['changer_mdp'], $_SESSION['idUser']));
    echo "<div class='alert'>Mot de passe changé</div>";
  }
  else {
    echo "<div class='alert'>Veuillez renseignez les deux champs avec le même
mot de passe</div>";
  }
}
else
{
    echo "<div class='alert'>Veuillez renseignez les deux champs pour changer
de mot de passe</div>";
}
```

```
?>
```

1.8 Utilisation de l'objet PDO

Pour assurer la connexion entre le site web et la base de donnée j'ai utilisé l'objet PDO, une extension php. En effet l'objet PDO, pour *PHP Data Object,* possède plusieurs méhodes permettant d'exécuter des requêtes SQL à partir du code PHP. Pour ce projet j'ai utilisé les méthodes *prepare()* et *execute()* ainsi que la chaîne de connexion à la base de donnée, appelée *Instance PDO.*

La méthode *prepare()* permet de préparer la requête SQL, donc comme paramètre une chaîne de caractère. L'utilisation de cette méthode permet la disposition de marqueur interrogatifs « ? » afin de désigner les valeurs à passées lors de l'exécution de la requête SQL.

La méthode *execute()* permet d'avoir comme paramètre un tableau contenant les valeurs qui viendront remplacer les marqueurs interrogatifs dans leur ordre de file.

1.9 Bilan personnel

Le point important que j'aimerai optimisé concerne l'instanciation PDO de la base de donnée. En effet dans toutes les pages de mon code on retrouve une connexion à la base de donnée, une phase redondante dont je peux m'en passer avec la création d'une page php que j'inclurai sur tout les pages. Cette page contiendra alors qu'une seule instance PDO.

Un autre point important serait de crypter le mot de passe des utilisateurs afin d'avoir une meilleure sécurité et de garder confidentiel les informations des utilisateurs. Ce cryptage se fera en hash MD5, avec les méthodes déjà implantées dans le php.

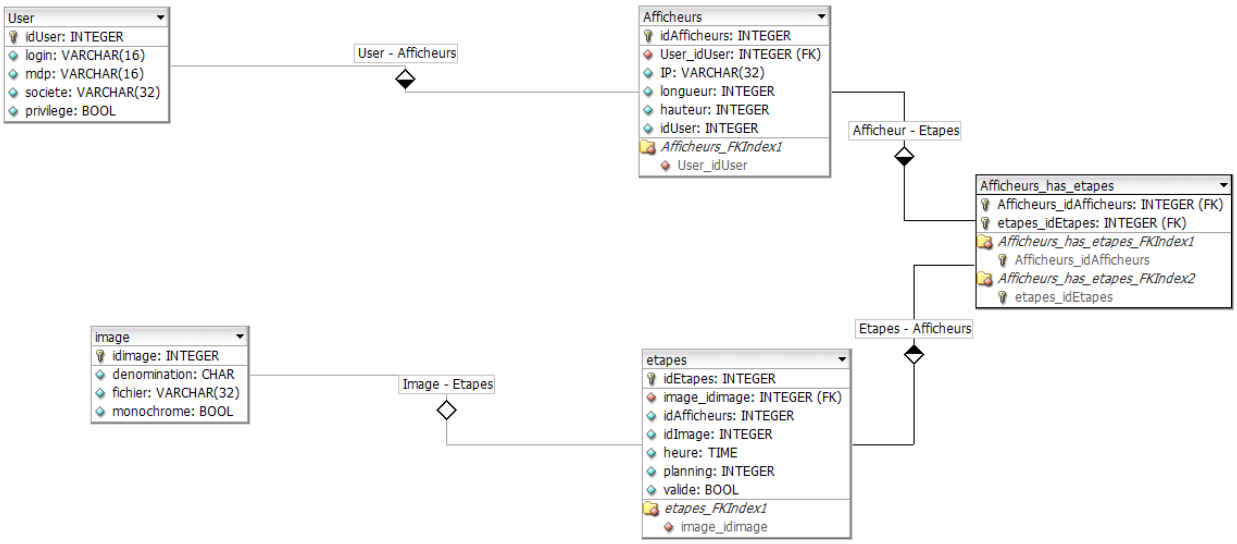
Pour finir, je souhaite effectuer plusieurs vérifications pour l'insertion d'une nouvelle image sur le site web. Démarche en cours de création lors de l'écriture de ce dossier.

5. Présentation de la partie «Affichage » de l'étudiant n°2

5.1. Base de données

L'objectif de ma tâche étant de traiter des images/textes et de l'envoyer sur la raspberry, je définirais cette tâche comme la jonction entre les travaux de mes camarades. En effet, j'ai travaillé sur la base de données de sa conception à sa réalisation. Celle-ci permet de stocker toutes les données entrées par un utilisateur dans les IHM du site web ainsi que les données utilisateurs tels que le login ou le mot de passe.
La structure de la base de données a était réalisé avec le logiciel DBDesigner.

Structure conceptuelle de la base de données :



Nous avons optimisé la base de données pour éviter les redondances d 'informations, limité l'espace de stockage utilisé éviter les erreurs. Ainsi dans cette base de données, nous avons :
- un utilisateur peut posséder un ou plusieurs afficheurs.
- un ou plusieurs afficheurs appartenant à un utilisateur.
- un afficheurs peut posséder plusieurs étapes.
- une ou plusieurs étapes peuvent appartenir à plusieurs afficheurs
- création d'une table qui fera la jonction entre la table étapes et afficheurs
 => afficheurs_has_etapes.
- une ou plusieurs étapes possèdent une ou plusieurs images.
- une image appartient à une étape.

Table user :

idUser	login	Mdp	Societe	privilège
Idantifiant	Nom d'utilisateur	Mots de passe définie par l'administrateur	Societe de l'utilisateur	Son rôle : Administrateur/utilisateur

Table afficheurs :

idAfficheurs	User_idUser	IP	longueur	hauteur
Identifiant	Clés étrangère sur le champs idUser de la table **user**	Adresse IP des raspberry	Longueur des panneaux leds installés	Hauteur des panneaux leds installés

Table etapes :

IdEtapes	image_idImage	heure	planning	valide
Identifiant	Clés étrangère sur le champ idImage de la table **image**	Heure à laquelle doit s'afficher l'étape	Identifiant permettant de regrouper plusieurs étapes dans un planning	Permet de savoir quelle étape est en cours d'affichage

Table images :

idImage	dénomination	Fichier	monochrome
Identifiant	Permet dans le récapitulatif de savoir si c'est une image ou du texte qui sera ou est affiché.	Nom du fichier	Booléen qui permet de savoir si l'image sera affiché en monochrome

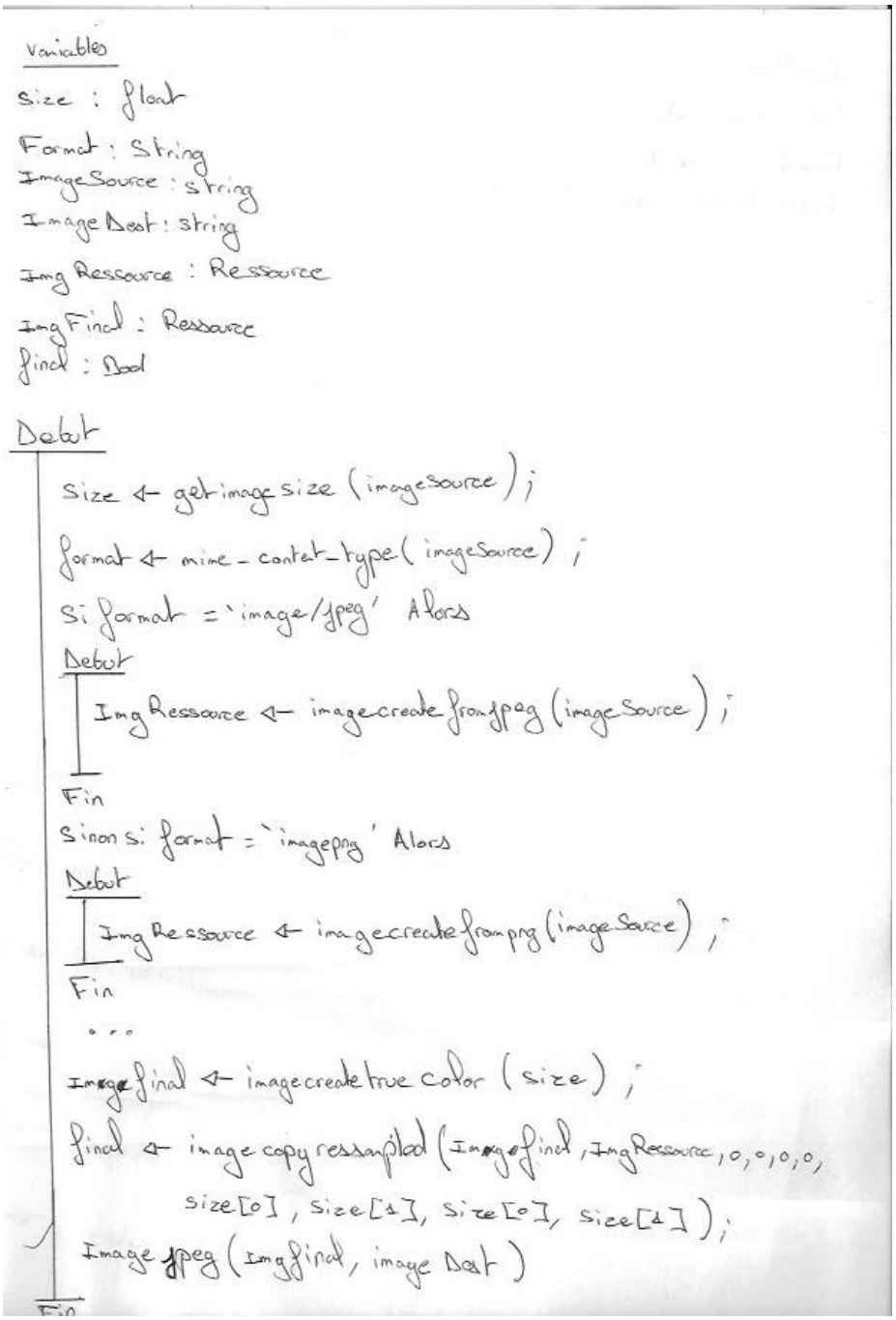
5.2. Programme de traitement d'image

Ce programme permet de traiter les images que l'utilisateur televerse sur le site en fonctions des options possibles demandé dans le cahier des charges et enregistre l'image dans le serveur web. Il est réalisé en PHP pour pouvoir intégrer le programme au site web facilement.
Dans les premières versions du programme, j'ai décidé de modifier la taille de l'image de sorte à ce quelle s'affiche sur les panneaux leds sans déformation. Je recadre donc l'image en fonction du nombre de panneaux leds présent. C'était une erreur de faire cela puisqu'avec le grand nombre de panneaux leds qu'il est possible de mettre l'algorithme était beaucoup trop long. De plus, nous nous sommes aperçu que l'une des librairies utilisé par l'étudiant n°3 sur la raspberry adapte déjà l'image en fonction du nombre de panneaux. Ainsi inutile de recadrer l'image.Il ne me reste donc plus que deux paramètres à ma fonction, la source de l'image et sa destination.
Une seul option dans le cahier des charges m'est demandé. Celle de pouvoir afficher l'image en monochrome si l'utilisateur le veut. Pour ce faire j'ai utilisé une librairie présente dans php pour traiter les images nommée GD. Dans cette librairie il existe une fonction permettant d'ajouter des filtres à nos images avec en premiers paramètre l'image à laquelle nous voulons appliquer le filtre et en second paramètre le filtre voulu. Pour nous ce sera donc le filtre « grayscale » .

```
imagefilter ( resource $image , int $filtertype )
```

source : www.php.net

5.2.1. Algorithme du programme :



```
Variables
Size : float
Format : String
ImageSource : string
ImageDest : string
ImgRessource : Ressource
ImgFinal : Ressource
final : Bool

Debut
    Size ← getimagesize (imageSource);
    format ← mime_content_type( imageSource) ;
    Si format = 'image/jpeg' Alors
    Debut
        ImgRessource ← imagecreatefromjpeg (imageSource) ;
    Fin
    Sinon si: format = 'imagepng' Alors
    Debut
        ImgRessource ← imagecreatefrompng (imageSource) ;
    Fin
    ...
    Imagefinal ← imagecreatetruecolor (size) ;
    final ← imagecopyressampled (ImageFinal, ImgRessource, 0, 0, 0, 0,
            size[0], size[1], size[0], size[1]);
    Imagejpeg (ImgFinal, imageDest)
Fin
```

5.2.2 Test unitaire de l’algorithme de la méthode

Action à tester	Tests à réaliser	Résultat(s) attendu(s)
Destination de l’image	**Envoyer l’image**	**L’image est dans le bon dossier**
Extension de l’image	**Sélectionner des images aux format jpeg, png, gif**	**L’image enregistrer est au même format que l’image sélectionné**
Taille de l’image	**Sélectionner une image trop volumineuse**	**Un message d’erreur indique que l’image sélectionné est trop volumineuse**

Pour accéder à la totalité des tests, dirigez vous dans la partie des tests d’intégrations.
Certains des tests unitaire ont été testé avec une base de données factice.

5.2.3. Code du programme :

```
function resize($imgSource, $imgDest)
{
 try{

  $bdd = new PDO('mysql:host=localhost;
 dbname=afficheur_publicitaire', 'jordan',
 'afficheur2019');
 }
 catch(exception $erreur){

  die('Erreur '.$erreur->getMessage());
 }

 $req = 'select fichier, monochrome, planning, heure, idAfficheur from
 image where image.idImage =    etape.idImage and etape.idUtilisateur =
 user.idUser";

 $size = getimagesize($imgSource); //recupere la taille de l'image
 originale

 $format = mime_content_type($imgSource); // permet de connaitre le
format  de l'image

 if($format == 'image/jpeg')
 {
  $imgRessource = imagecreatefromjpeg($imgSource); //copie les
 caractéristique de l'image     originale
 }
 else if($format == 'image/png')
 {
  $imgRessource = imagecreatefrompng($imgSource);
 }
 else
 {
  $imgRessource = imagecreatefromgif($imgSource);
 }

 $imgFinal = imagecreatetruecolor($size[0], $size[1]); //recréer
l'image       avec les nouvelles dimensions

 $final = imagecopyresampled($imgFinal, $imgRessource, 0, 0, 0, 0,
 $size[0], $size[1], $size[0],  $size[1]); //size[0] = width, size[1]
 = height

 if($format == 'image/jpeg')
 {
  imagejpeg($imgFinal, $imgDest); //Creation physique
 }
 else if($format == 'image/png')
 {
```

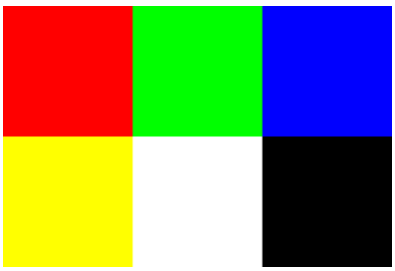
```
 imagepng($imgFinal, $imgDest); //Creation physique
}
else
{
 imagegif($imgFinal, $imgDest); //Creation physique
}
}
```

5.3. Programme de traitement de texte

Ce programme permet de convertir le texte insérer par l'utilisateur et de le convertir en image afin de pouvoir être affiché sur les panneaux leds. Au départ j'ai appris que les panneaux étaient capable de lire le format .ppm qui est un format de fichier graphique. Cependant ce format de fichier est codé d'une façon difficilement associable à notre projet.

Exemple :

```
255   0    0      0   255   0         0    0   255
255 255    0    255 255 255         0    0     0
```

Résultat : 

source : https://fr.wikipedia.org/wiki/Portable_pixmap

Plus tard en retournant jeté un œil sur la librairie GD je vis qu'il était possible d'insérer du texte dans une image grâce à une fonction.

```
imagettftext ( resource $image , float $size , float $angle , int $x , int $y ,
int $color , string $fontfile , string$text ) : array
```

source : www.php.net

Dans ce programme nous n'avons pas de valeur initiale pour la taille de l'image. Ainsi cette fois-ci il va falloir calculer la taille de l'image en fonction du nombre de panneaux.
Tout d'abord, je dois récupérer le nombre de panneaux que l'utilisateur possèdent grâce à une requête SQL

```
$taillePanneaux = $bdd -> prepare(select hauteur, longueur from
afficheur where user_idUser = ?);
$taillePanneaux -> execute(array($session['idUser']));
```

J'initialise une taille correspondant à une panneau. Un panneau étant carré d'une taille de 32x32 je l'initialise à 128x128. Ensuite, je fais deux boucles « for » dans laquelle j'incrémente la taille de 128/panneaux. Pourquoi cette taille ? Parce qu'elle n'est ni trop petite ni trop grande par rapport aux nombre maximum de panneaux que l'on peut mettre sur la raspberry soit 12x3. Pour 12x3 panneux on aurait une résolution d'image de 1536x384.

```
$height = 128;
$width = 128;

for($i=0; $i<$taillePanneaux['hauteur']; i++){
 $height+128;
 }
 for($i=0; $i<$taillePanneaux['longueur']; i++){
 $width+128;
 }
```

Ensuite, comme pour l'image classique je crée l'image avec les caractéristiques précédemment indiqué.

```
$img = imagecreatetruecolor($width,$height);
```

Pour la couleur du texte, l'utilisateur la choisi à partir d'une palette de couleur insérer sur le site en HTML. Cette palette de couleur retourne une valeur en hexadécimal que je récupère en POST. Cependant, la fonction qui permet d'associer une couleur à un texte dans la librairie GD doit être sous la forme RGB.

```
imagecolorallocate ( resource $image , int $red , int $green , int $blue ) : in
t
```

source : www.php.net

Ainsi j'utilise la fonction « List » pour ranger dans trois variables ma valeur hexadécimale découpé en trois.

```
list($r, $g, $b) = sscanf($Txt_color, "#%02x%02x%02x"); //
Transformation hexa -> 3 couleurs red, green, blue

$color = imagecolorallocate($img, $r, $g, $b);
```

Je crée ensuite une variable avec les valeurs de la couleur blanche et je la mets en fond de l'image grâce à une fonction associée. J'utilise ensuite la fonction décrite plus haute pour ajouter le texte à l'image et enfin je crée l'image au format png puisque le format est plus adapté pour le texte.

```
$white = imagecolorallocate($img, 255, 255, 255);
imagefilledrectangle($img, 0, 0, $width, $height, $white); //Fond de
l'image
$font = 'C:\Windows\Fonts\arial.ttf';

imagettftext($img, 30, 0, $width/3, $height/2, $color, $font, $Txt);
//Dessine le texte avec en parametre la taille de la police, la
position, la couleur, le type de police et le texte.

imagepng($img, $chemin); //Création de limage au format png, le format
png rend le texte plus nette

imagedestroy($img);
```

5.3.1. Test unitaire de l'algorithme de la méthode

Dans le test unitaire on peut pas choisir la couleur du texte il faudra attendre l'intégration.

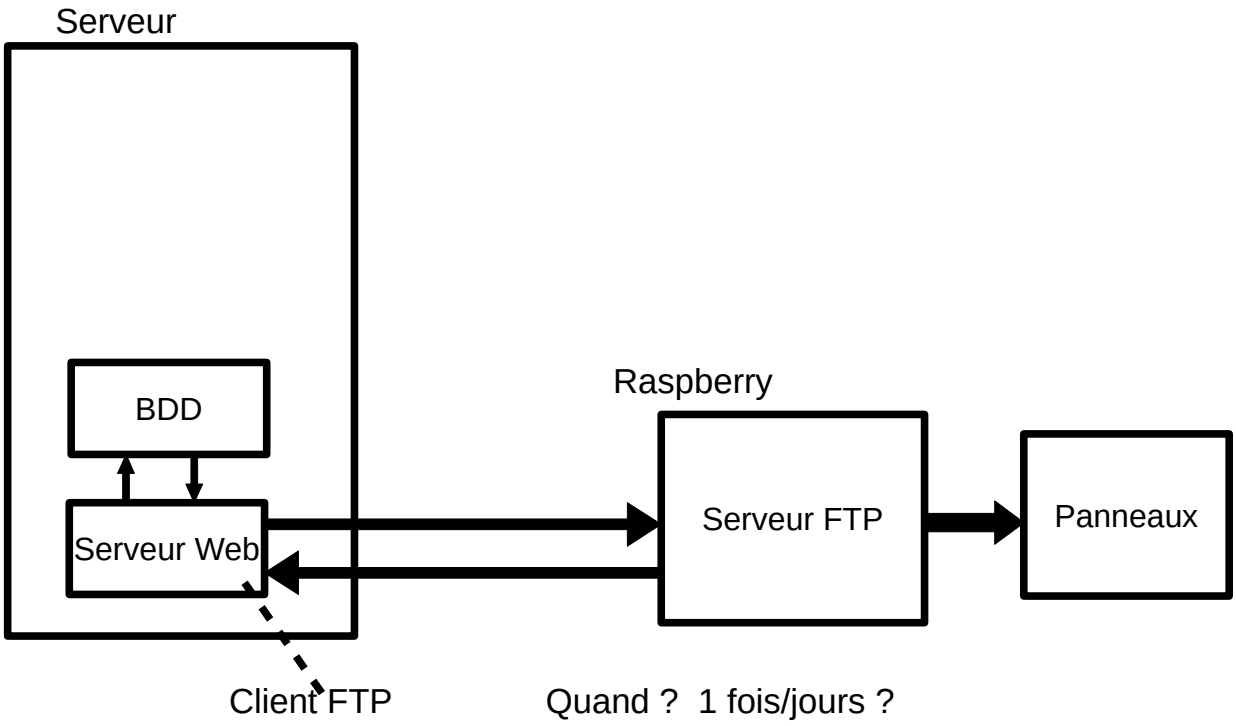
Action à tester	Tests à réaliser	Résultat(s) attendu(s)
Destination de l'image	**Envoyer le texte**	**L'image est dans le bon dossier**
Texte sur l'image	**Envoyer le texte**	**Le texte est présent sur l'image**
Centrage du texte	**Envoyer le texte**	**Le texte est centrer sur l'image**
Format de l'image	**Envoyer le texte**	**L'image est au format png**
Fond de l'image	**Envoyer le texte**	**Le fond de l'image est blanc**

5.4. Programme de génération de fichier XML

Dans la page récapitulatif du site web, un bouton « afficher » permet d'afficher la planning sélectionné sur les panneaux leds. Pour ce faire un fichier XML est généré avec les informations nécessaire à mon collègue qui travaille sur la raspberry. Le fichier XML et les images sont ensuite envoyer par FTP.

Il a fallut se questionner sur quand est-ce que l'on allait envoyer le planning de l'utilisateur sur la Raspberry. Nous avions au départ opté pour que le planning soit envoyer chaque jour à 7h mais cela posais beaucoup de problème. En effet si l'utilisateur modifie son son affichage au cours de la journée, le planning ne sera mis a jour que le lendemain à 7h. De plus, si la raspberry ou les afficheurs sont hors tension lors de l'envoi, il n'y aurais pas de mis a jour de faites. Nous avons donc ensuite

penser à l'envoyer tout les X temps mais cela ferais beaucoup trop de requête pour le serveur surtout si on imagine qu'il y ait des centaine d'afficheurs connecté à celui-ci. Nous avons donc finalement opté par un envoie par l'utilisateur. C'est à dire lorsqu'il le souhaite.



5.4.1. Code du programme XML

```
<?php
 $xml = '<?xml version="1.0" encoding="UTF-8"?>';
 $xml .= '<Affichage>';

 $bdd = new
 PDO('mysql:host=localhost;dbname=afficheur_publicitaire;charset=UTF8',
'jordan',     'afficheur2019');
 $info = $bdd -> prepare("SELECT fichier,heure,hauteur,longueur FROM
afficheur,    etapes, image WHERE planning = ? AND user_idUser=?");
 $info -> execute(array($Planning, $_SESSION['idUser']));

 while($data = $info -> fetch(PDO::FETCH_ASSOC)){ //Tant qu'il y a des
données       on reste dans la boucle.
  $xml .= '<image><fichier>'.$data['fichier'].'</fichier>';
  $xml .='<heure>' .$data['heure'].'</heure>';
  $xml .='<hauteur>' .$data['hauteur'].'</hauteur>';
  $xml .='<longueur>' .$data['longueur'].'</longueur></image>';
 }

 $xml .= '</Affichage>';

 file_put_contents('Affichage.xml', $xml); //Ecriture dans un fichier.
?>
```

5.2.2. Test unitaire de l'algorithme de la méthode

Action à tester	Tests à réaliser	Résultat(s) attendu(s)
Destination du fichier	**Appuyer sur « afficher »**	**Le fichier XML est envoyer par FTP**
Contenu du fichier	**Ajouter plusieurs étapes et afficheurs dans la base de données et générer le fichier XML**	**Le fichier XML contient toutes les informations requise filtrer par utilisateur et planning**

5.4.3. Résultat du fichier XML

5.5. Programme de communication FTP

Dans un premier temps nous avons eu l'idée d'envoyer les données par TCP. Cependant, ce protocole n'était pas adapté à notre projet. En effet nous envoyons des images de tailles variables pouvant atteindre quelques mégas octets, il faudrait donc découper l'image en plusieurs paquets pour ensuite une fois les paquets présents sur le serveur les rassembler. D'autant plus que généralement il sera nécessaire d'envoyer plusieurs images à la fois pour avoir un planning complet.
Nous avons donc opter pour une solution plus simple et plus adapté c'est à dire une communication Client/Serveur par FTP. J'ai donc installé un serveur FTP sur la Raspberry (voir dossier d'installation du système). À noter que pour un serveur FTP distant il est nécessaire de configurer les ports de son routeur afin d'autoriser la communication.

Exemple pour un routeur orange :

Configuration NAT/PAT

Les règles NAT/PAT sont nécessaires pour autoriser une communication initiée depuis Internet pour atteindre un appareil spécifique de votre réseau. Vous pouvez aussi définir le(s) port(s) sur lequel cette communication sera acheminée.

NB : les règles NAT/PAT suivantes s'appliquent uniquement à IPv4.

Assurez-vous de ne pas avoir filtré ces ports dans le pare-feu

Règles personnalisées

application / service	port interne	port externe	protocole	appareil	activer	
Secure SI	49349	49349	les deux	SIMON-F		enregistrer
FTP Server	21	21	les deux	Simon-Server	☑	supprimer
FTP Data	20	20	TCP	Simon-Server	☑	supprimer

5.5.1. Code du programme FTP

```php
<?php

if(isset($_POST['envoyer']))
{
 $ftp_server = "169.254.26.48";
 $ftp_user_name="pi";
 $ftp_user_pass="raspberry";
 $source_file = $_FILES["fichier"]["name"];
 $source_files = $_FILES["fichiers"]["name"];
 $destination_file = "~/Fichiers/image30.jpg";
 $destination_files = "~/Fichiers/image22.jpg";

 // Création de la connexion
 $conn_id = ftp_connect($ftp_server);

 // Authentification avec nom de compte et mot de passe
 $login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

 // Vérification de la connexion
 if ((!$conn_id) || (!$login_result)) {
  echo "La connexion FTP a échoué!";
  echo "Tentative de connexion à $ftp_server avec $ftp_user_name";
  die;
  }
  else{
  echo "Connecté à $ftp_server, avec $ftp_user_name";
  }

 // Téléchargement d'un fichier.
 $upload = ftp_put($conn_id, $destination_file, $source_file,
FTP_ASCII);
 $upload = ftp_put($conn_id, $destination_files, $source_files,
FTP_ASCII);
 // Vérification de téléchargement
 if (!$upload) {
  echo "Le téléchargement Ftp a échoué!";
  } else {
  echo "Téléchargement de $source_file sur $ftp_server en
$destination_file";
  }

 // Fermeture de la connexion FTP.
 ftp_quit($conn_id);

}
?>
```

5.5.2. Test unitaire de l'algorithme du programme

Action à tester	Tests à réaliser	Résultat(s) attendu(s)
Connexion au serveur FTP	**Aller sur la page contenant le code de la communication FTP**	**Aucune erreur de connexion**
Envoie des fichiers	**Sélectionner une image et appuyer sur un bouton envoyer**	**Les fichiers images et XML sont envoyer sur la Raspberry**

5.6. Bilan personnelle

5.6.1. Planning réalisé

Je n'ai pas réellement respecté les temps du planning prévisionnel que j'ai réalisé avec mindView puisque parfois il a fallut revenir en arrière pour refaire la base de données

6. Présentation de la partie «Affichage » de l'étudiant n°2

Cette tâche consiste à réaliser la commande de l'afficheur (piloter l'afficheur) et les communications (Communiquer) tel que l'installation du réseau et l'installation des afficheurs.
Cette tâche sera essentiellement autour de l'afficheur, pour le faire afficher les images reçues qui sont contenues dans un planning et les paramétrer pour faire correspondre la disposition des panneaux d'affichage. Lors de la réception du fichier de configuration qui sera sous forme XML qui sera créé à partir du site web à l'aide de la Base de Données pour les remplir les informations, le programme obtiendra les paramètres qu'il lui sera nécessaire pour afficher correctement les images, de plus il reçoit le planning qui contiendra les noms des images avec l'horaire de passage. L'afficheur doit pouvoir passer sur un nouveau planning si jamais il reçoit un nouveau fichier XML.

6.1 Matériel

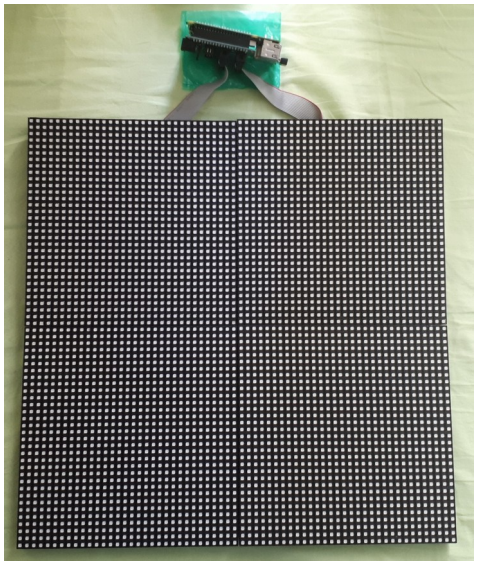
Pour cette partie différents matériels ont été utilisé pour permettre l'affichage, le matériel qui a été utilisé sont les suivants :

- Des panneaux d'affichage LED 32x32
- Des câbles plats pour les port HUB75
- Une alimentation 5v et ses câbles d'alimentation pour la Raspberry PI 3 et les panneaux LED
- Une Raspberry PI 3
- Une carte add-on HAT-A3

6.1.1 Panneaux d'affichage LED 32x32

- 32x32 LED -> 1024 LED au Total
- Scan rate : 1/8
- Voltage : 5V
- taille : 192x192 mm
- Fréquence : 60Hz/s
- Taux de Rafraichissement : ~2000Hz
- 256 niveaux de chaque couleurs
- Consommation énergétique moyenne : 400~450 W/m²

Panneau fournis pour le projet :

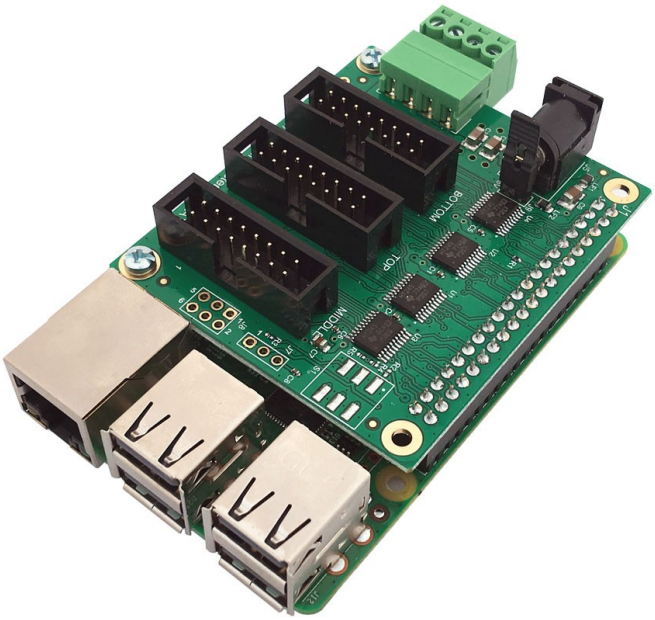


6.1.3 Raspberry PI 3

-SoC / « Système sur une puce »: Broadcom BCM2837
-Processeur: 4× ARM Cortex-A53, 1.2GHz
-Processeur graphique: Broadcom VideoCore IV
-RAM: 1GB LPDDR2 (900 MHz)
-Réseau: 10/100 Ethernet, 2.4GHz 802.11n wireless
-Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy
-Espace de stockage: microSD
-GPIO: 40-pin header, populated
-Ports: HDMI, 3.5mm analogue audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)
-Voltage : 3,3V~5V

Nano ordinateur utilisé pour la mise en fonction des panneaux LED. Remplaçant de l'ancienne solution possible (ESP32) pour des raisons de performance, sur le fait du nombre de panneaux LED que peux supporter l'ESP32 (2 afficheurs vs 36 afficheurs pour la Raspberry PI 3 avec une Carte add-on), de plus l'ESP 32 pouvait supporter soit une carte SD ou 2 afficheurs, alors ce qui nous a amené au Raspberry PI 3. De plus j'ai installé un package pour que le ftp fonctionne sur la Raspberry PI 3, ce qui sera le moyen de recevoir les images.

6.1.4 Carte add-on HAT-A3

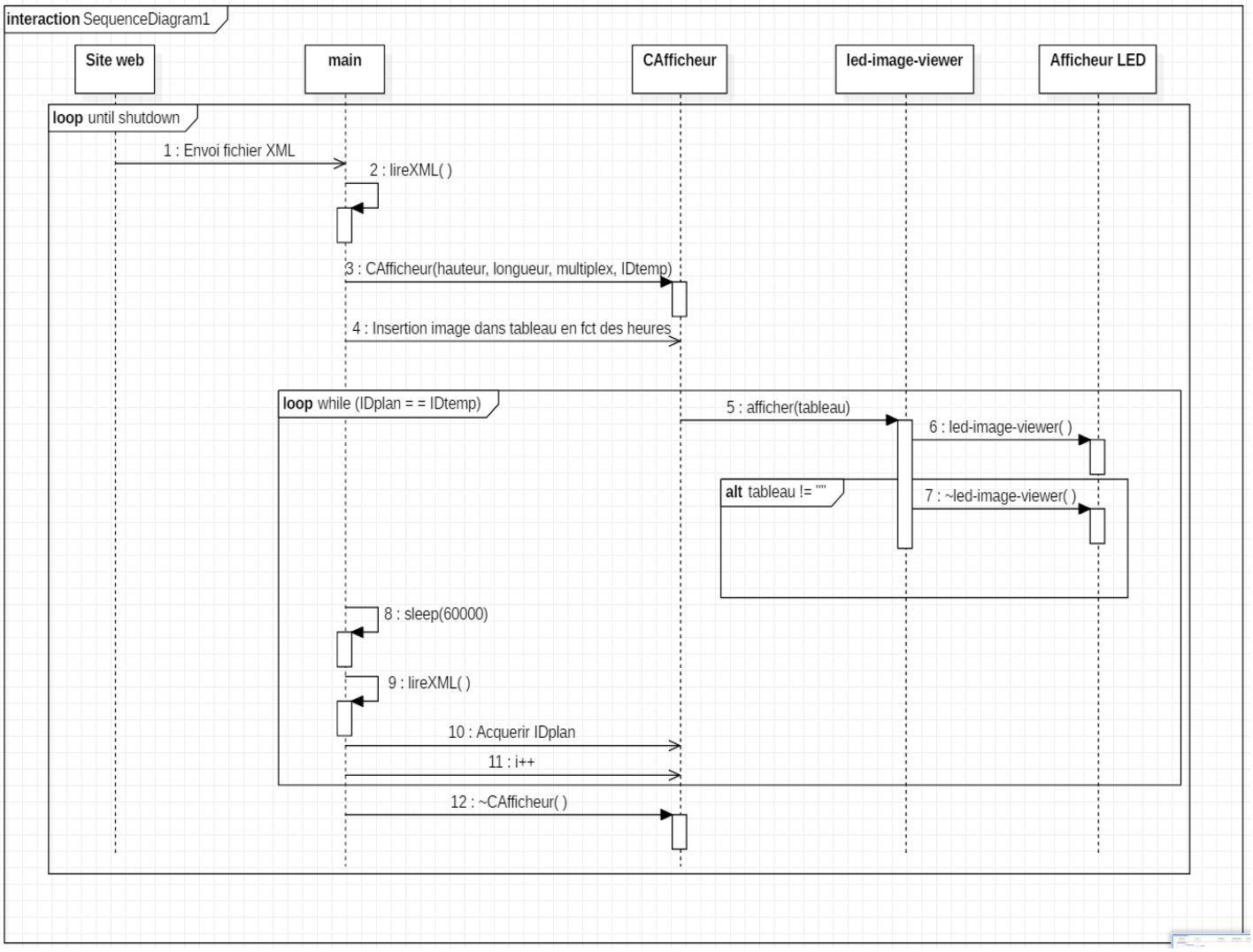


- Carte Active-3 designée par Henner Zeller créateur de la bibliothèque rpi-rgb-led-matrix
- Supporte 3*12 panneaux LED 32x32
- Alimente la Raspberry PI 3 en 5VDC si utilisation d'une alimentation pour les panneaux LED

Faite pour la bibliothèque rpi-rgb-led-matrix donc contrôle les panneaux LED avec l'aide de la Raspberry PI 3 et solution pour le problème cité auparavant.

6.2.1 Diagramme de séquence

Diagramme de séquence du programme de commande d'affichage



Le diagramme de séquence au dessus explique le fonctionnement du programme, le programme reçoit un fichier XML que le programme lira grâce à une bibliothèque nommée « tinyxml2 », ensuite il instancie CAfficher() qui utilisera la bibliothèque « rpi-rgb-led-matrix » pour afficher des images sur les panneaux d'affichage. Une boucle est mise en place pour pouvoir changer de planning lorsque l'ID du nouveau planning est différent du planning actuel. Après si jamais le le planning change, le programme ce ré-exécute depuis le début.

6.2.2 Présentation librairie tinyxml2

Cette bibliothèque est un analyseur XML, il analyse les fichiers XML et les convertis sous un format lisible pour le programme, ce qui permet la lecture,

l'édition et la sauvegarde. Mais dans ce cas, ce qui nous intéresse c'est la lecture.

Code :

XML reçu :

```
<planning>
<image>
 <fichier>test.png</fichier>
 <heure>07:00:00</heure>
 <hauteur>2</hauteur>
 <longueur>2</longueur>
</image>
<id>1</id>
</planning>
```

Extraction de l'ID grâce à tinyxml2

```
#include <iostream>
#include <tinyxml2.h>
main()
{
XMLDocument plan; //instanciation de la méthode
plan.LoadFile( "planning.xml" );   //charger xml
//introduction de la valeur lue dans IDplan
XMLText* textNode = plan.FirstChildElement(
"planning" )->FirstChildElement( "id" )-
>FirstChild()->ToText();
IDplan = textNode->Value();
}
```

On reçoit donc un XML dans lequel on aimerait récupérer l'ID du planning, pour ce faire, avec tinyxml2 on va chercher dans la balise <id> l'ID du planning.

6.2.3 Présentation librairie time.h

On utilise un planning qui permet au gérant du magasin de programmer plusieurs affichages pendant une journée, pour ce faire j'ai du utiliser la bibliothèque time.h pour pouvoir afficher les images au bon moment avec les heures écrites dans le fichier XML.

Code :

Conversion du temps récupéré en XML vers time_t

```
#include <iostream>
#include <tinyxml2.h>
main()
{
const char *heure = "15:25:00";
struct tm temps; //Structure du temps
//écriture du temps dans la structure
strptime(heure, "%H:%M:%S", &temps);
time_t heure_t = mktime(&temps);  //Conversion
vers time_t
}
```

Avec cette bibliothèque il y a une variable « time_t » qui le temps en secondes depuis le 01/01/1970 à 00:00:00, on utilise du coup cette valeur pour intégrer les images aux heures qu'il faudra. De plus la « struct tm » permet de créer une structure qui contient les différent temps, c'est-à-dire :

tm_sec	int	secondes après les minutes	0-61*
tm_min	int	Minutes après les heures	0-59
tm_hour	int	Heures depuis minuit	0-23
tm_mday	int	Jours du mois	1-31
tm_mon	int	Mois depuis Janvier	0-11
tm_year	int	Années depuis 1900	
tm_wday	int	Jours depuis Dimanche	0-6
tm_yday	int	Jours depuis le 1er Janvier	0-365
tm_isdst	int	Heure d'été	

Les valeurs enfin récupérées on range ceci dans un tableau qui sera affilié à une image pour le planning.

6.2.4 Présentation librairie rpi-rgb-led-matrix

Pour l’affichage des image, cette bibliothèque à été faite pour la carte HAT-A3. Elle permet à la carte HAT-A3 de contrôler les affichages. La bibliothèque inclus des applications liées qui permettent l’affichage avec la configuration en paramètre à la commande.

Code :

« sudo ./led-image-viewer --led-chain=1 --led-parallel=1 image.jpg »

Lors de l'entrée de cette ligne sur la console, l'image sera affiché sur l'afficheur LED qui sera branché sur la prise par défaut et elle sera longue d'un afficheur, et le nom de l'image est « image.jpg ».
Plusieurs paramètres sont disponibles :

--led-gpio-mapping=<gpio-mapping> : Nom du mappage GPIO. Par défaut "regular"

--led-rows=<rows> : Nombre de LED par rangée. 8, 16, 32 or 64. (Par défaut: 32).

--led-cols=<cols> : Nombre de LED par colonnes. 32 or 64. (Par défaut: 32).

--led-chain=<chained> : Nombre de panneaux chaînés. (Par défaut: 1).

--led-parallel=<parallel>: Nombre de panneaux parallèle. 1..3 (Par défaut: 1).

--led-multiplexing=<0..4> : Multiplexage pour le scan-rate type: 0=direct; 1=strip; *2=checker; 3=spiral; 4=Z-strip (Par défaut: 0)*

--led-brightness=<percent> : Luminosité en pourcent (Par défaut: 100).

Pour choisir le scan rate ce tableau est présent :

Type w*h	**Scan Multiplexing / Scan Rate**	**Program commandline flags**
64x64	1:32	--led-rows=64 --led-cols=64
64x64	1:32	--led-rows=64 --led-cols=64 --led-row-addr-type=1
64x32	1:16	--led-rows=32 --led-cols=64
64x32	1:8	--led-rows=32 --led-cols=64 --led-multiplexing=1
32x32	1:16	--led-rows=32
32x32	1:8	--led-rows=32 --led-multiplexing=1
32x16	1:8	--led-rows=16
32x16	1:4	--led-rows=16 --led-multiplexing=1
32x16	1:4	--led-rows=16 --led-row-addr-type=2 --led-multiplexing=4

Pour nos panneaux on aura :

« sudo ./led-image-viewer --led-chain=3 --led-parallel=1 --led-multiplexing=1 image.jpg »

6.2.5 Classe CAfficheur()

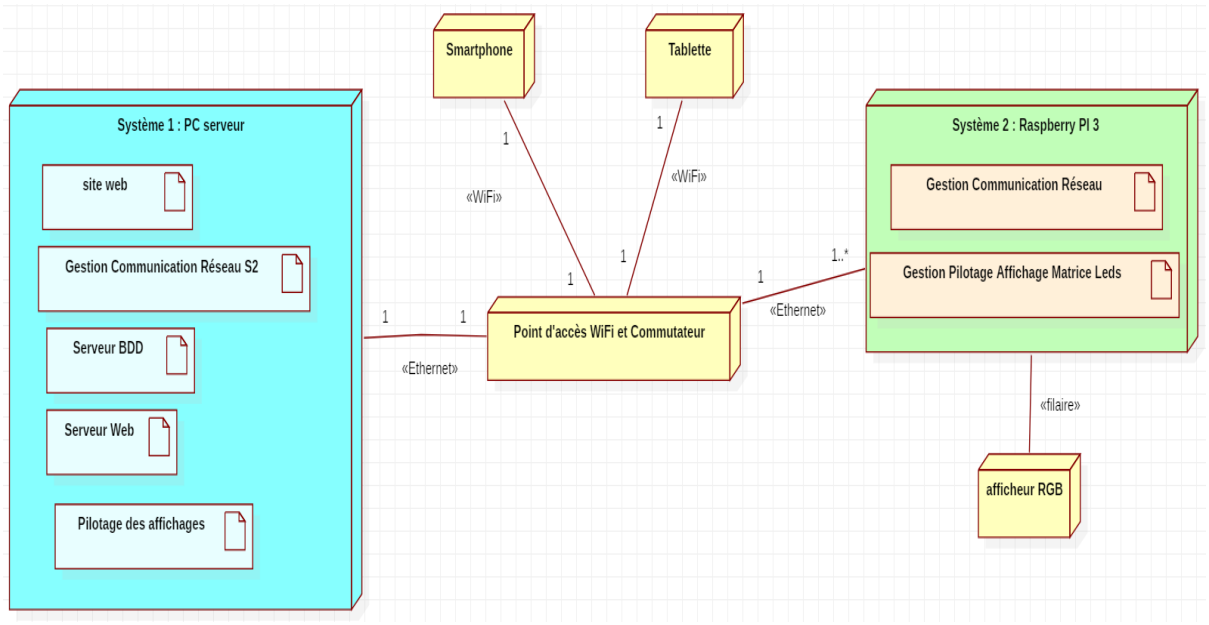
Cette classe à été crée pour utiliser les différentes bibliothèques cité jusqu'à présent avec pour Constructeur :
« CAfficheur::CAfficheur(entier hauteur,entier longueur,entier multiplex,entier IDtemp) »

Pour Destructeur :
« CAfficheur::~CAfficheur() »

Et comme méthodes :
« néant CAfficheur::Afficher(chaine de caractères img) » Pour afficher les images grâce à la librairie rpi-rgb-led-matrix.
« néant CAfficheur::LireXML() » Pour lire le XML reçu grâce à tinyxml2.

6.2.6 Réseau

La Raspberry PI 3 sera connectée en Wi-Fi sur le réseau local, je modifie des paramètres dans la box pour pouvoir attribuer une IP fixe sur la Raspberry PI 3. Grâce à ça, si les ports FTP pour la Raspberry PI 3 sont ouvert, il peux recevoir des fichiers dans le dossier désigné.



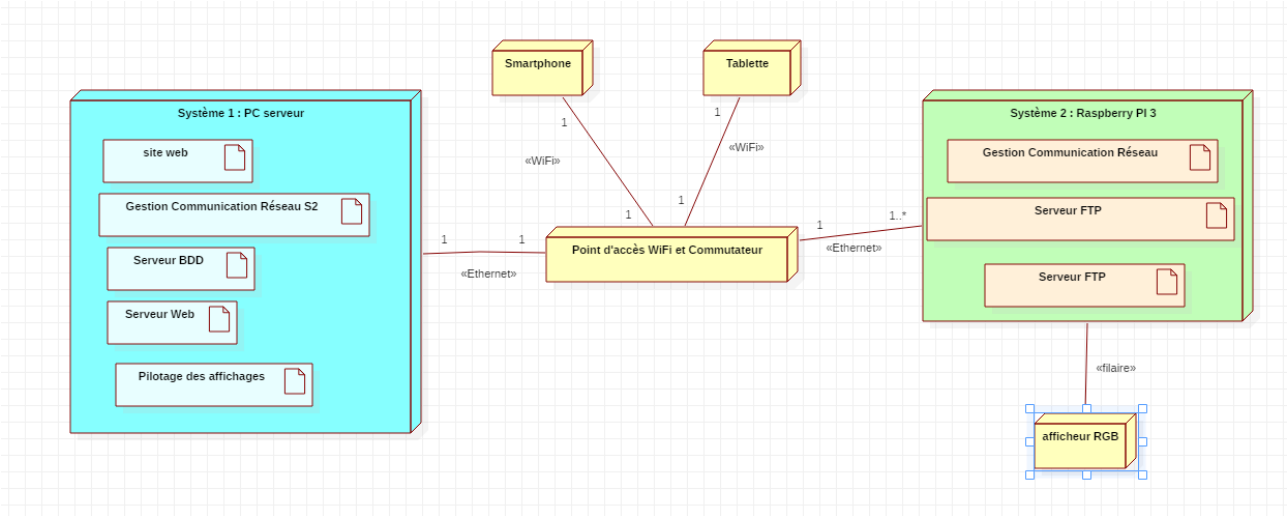
6.2.7 Plan de test unitaire

Action à vérifier	Tests à effectuer	Résultats attendus	Résultat obtenu
Commander afficheur	Configurer l'afficheur et afficher les images reçues	Afficheur configuré et affiche les images reçues	L'afficheur est bien configuré et affiche les images reçues.
Communication entre PC et raspberry	Envoi de fichiers vers la Raspberry	Le fichier reçu sur la Raspberry	Le fichier est reçu sur la raspberry et est lu par le programme.

7. Intégration et test

7.1. Intégration

Diagramme de déploiement final :



7.2. Test d'intégration

7.2.1. XML

Action à tester	Tests à réaliser	Résultat(s) attendu(s)	Résultat(s) obtenu(s)	Remède(s)
Destination du fichier	**Appuyer sur « afficher »**	**Le fichier XML est envoyer par FTP**	**Le fichier XML est envoyer par FTP**	
Contenu du fichier	**Ajouter plusieurs étapes et afficheurs dans la base de données et appuyer sur afficher puis vérifier le contenu du fichier XML**	**Le fichier XML contient toutes les informations requise filtrer par utilisateur et planning**	**Le fichier XML contient toutes les informations requise filtrer par utilisateur et planning**	

7.2.2. FTP

Action à tester	Tests à réaliser	Résultat(s) attendu(s)	Résultat(s) obtenu(s)	Remède(s)
Connexion au serveur FTP	**Aller sur la page contenant le code de la communication FTP**	**Aucune erreur de connexion**	**Erreur**	**Les identifiant et mots de passe sont ceux de la connexion à l'OS de la raspberry**
Envoie des fichiers	**Sélectionner une image et appuyer sur un bouton envoyer**	**Les fichiers images et XML sont envoyer sur la Raspberry**	**Les fichiers images et XML sont envoyer sur la Raspberry**	

7.2.3. Texte

Action à tester	Tests à réaliser	Résultat(s) attendu(s)	Résultat(s) obtenu(s)	Remède(s)
Destination de l'image	**Envoyer le texte**	**L'image est dans le bon dossier**	**L'image est dans le bon dossier**	
Texte sur l'image	**Envoyer le texte**	**Le texte est présent sur l'image**	**Le texte est présent sur l'image**	
Couleur sur le texte	**Sélectionner une couleur**	**Le texte possède la couleur sélectionné par l'utilisateur**	**Le texte possède la couleur sélectionné par l'utilisateur**	
Centrage du texte	**Envoyer le texte**	**Le texte est centrer sur l'image**	**Texte mal centré**	**Calcul afin de centrer le texte correctement**
Dimension de l'image	**Sélectionner différents afficheurs et envoyer le texte**	**Les dimension de l'image s'adapte en fonction du nombre de panneaux présent sur un afficheur**		
Format de l'image	**Envoyer le texte**	**L'image est au format png**	**L'image est au format png**	
Fond de l'image	**Envoyer le texte**	**Le fond de l'image est blanc**	**Le fond de l'image est blanc**	

7.2.4. Image

Action à tester	Tests à réaliser	Résultat(s) attendu(s)	Résultat(s) obtenu(s)	Remède(s)
Appliquer un filtre à l'image	**Cocher l'option « monochrome »**	**l'image est monochrome**	**l'image est monochrome**	
Destination de l'image	**Envoyer l'image**	**L'image est dans le bon dossier**	**L'image est dans le bon dossier**	
Extension de l'image	**Sélectionner des images aux format jpeg, png, gif**	**L'image enregistrer est au même format que l'image sélectionné**	**L'image enregistrer est au même format que l'image sélectionné**	
Taille de l'image	**Sélectionner une image trop volumineuse**	**Un message d'erreur indique que l'image sélectionné est trop volumineuse**	**Code à réaliser**	
Type de fichier sélectionné	**Essayer de sélectionner un fichier autre qu'une image**	**Impossibilité de sélectionné d'autre type que les formats jpeg, png et gif**	**Impossibilité de sélectionné d'autre type que les formats jpeg, png et gif**	

7.2.5. Authentification

Action à tester	Tests à réaliser	Résultat(s) attendu(s)	Résultat(s) obtenu(s)	Remède(s)
Connecter l'utilisateur	Remplir le formulaire avec des informations valides	L'utilisateur est connecté	L'utilisateur est bien connecté sur sa session	
Connecter l'utilisateur avec des informations incorrectes	Remplir le formulaire avec des informations invalides	La page indique que les identifiants sont faux	La page refuse l'accès est indique la bonne erreur	
Connecter l'utilisateur avec un champ vide	Remplir qu'un seul champ du formulaire	La page indique que l'un des champs est manquant	La page refuse l'accès est indique la bonne erreur	
Accéder à la page d'accueil sans connexion	Saisir l'URL de la page dans le navigateur	Redirection sur la page de connexion	Redirection automatique sur la page de conexion	
Aucun afficheur défini	Première connexion	Redirection sur la page d'ajout d'afficheur	Redirection automatique sur la page de création d'afficheur	
Un ou plusieurs afficheur(s) défini(s)	Compte possédant déjà des afficheurs	Accès à la page d'accueil	l'utilisateur accède à la page d'accueil	

8. Bilan technique

8.1. Critiques du projet

Si c'était à refaire, nous serions passé à l'intégration bien plus rapidement parce que l'intégration génère beaucoup d'erreurs qui demande beaucoup de temps pour être pallié.
Nous aurions réalisé une base de données définitive et concrète plus rapidement également ce qui nous aurai évité de réaliser les requête uniquement lors de l'intégration.
En ce qui concerne l'organisation du site, nous avons rangé les divers fichiers que récemment, nous avons donc travailler dans un environnement de test longtemps. Et, d'une manière général, nous avons travailler une bonne partie du projet comme si nous étions en phase de test.
Le manque d'organisation en l'enceinte de l'équipe a été ce qui a mis le groupe en retard sur certains points.

8.2. Evolution possible

Avec le matériel que nous avons, il est possible de faire passer des vidéos sur les panneaux leds. Ainsi, la porte est ouverte à une évolution de notre projet pour intégrer les vidéos.
Le micro-ordinateur Raspberry pi 3 que nous utilisons possède des caractéristiques techniques au-delà de nos nécessités. Ainsi utiliser un autre micro-ordinateur/processeur plus petit serais économique.
Nous avons pensé à installer un système de protection en plastique ou autres autour de la Raspberry avec un système de ventilation afin de la protéger du climat en cas de mise en place en extérieur.

8.3. Échanges avec les industriels

Au tout début, le projet était prévu pour être réalisé avec le micro-processeur ESP32 cependant celui-ci ne convenais pas au projet. Au début du projet nous ne pouvions pas savoir si nous allions avoir besoin d'utiliser du temps réel et il valait mieux prévoir que c'était nécessaire, or ce micro-processeur n'avait pas un réel noyau temps réel. De plus, il y n'y avait pas assez de broches sur la carte et donc aucun moyen d'ajouter un espace de stockage non présent initialement sur celle-ci. Alors, nous avons échangé avec M.Bellouet sur le choix d'un nouveau micro-processeur et nous avons opter pour la Raspberry.

Site internet d'EBconnections : http://www.ebconnections.com/

9. Sources

Carte HAT-A3 : https://www.acmesystems.it/HAT-A3
Panneaux LED : https://www.acmesystems.it/ledpanel
php.net