



Groupe BMC

2021-2022



ThalesAlenia
a Thales / Leonardo company Space

RAPPORT

SAE 24 & 23

PARTICIPANTS :

FARETIE BENJAMIN
PERRE MATTHIEU
WOODMAN CLARA

ORGANISÉ PAR :

Université Côte d'Azur
& ThalesAlenia Space

I. Création du projet par partie (cf.2)

1. PLANIFICATION DU PROJET

1.1. Répartition des tâches	2
1.2. Répartition des tâches	3
1.3. Diagramme de Gantt	4

2. BASE DE DONNEES

2.1. Définir la base de données	4
2.1.1. Diagramme entité association	4
2.2. Optimisation de la base de données	5
2.3. Exécution de la base données (Test)	5
2.4. Accessibilité de l'administration	7
2.5. Liaison avec le site Web (autonome)	7

3. PROGRAMMATION

3.1. Réaliser le programme principal	8
3.2. Définir les fonctions	8
3.2.1. Fonction « PrisePhoto »	9
3.2.2. Fonctions « Serveur/Client »	9
3.2.3. Fonction « AllumerLED »	10
3.2.4. Mise à jour de la base de données	10
3.3. Prise de photo depuis le site Web	11

4. SITE WEB

4. Maquette de fonctionnement du site	12
4.1. Réalisation de la page de connexion	13
4.1.1. Design et mise en page	13
4.1.2. Fonctionnement pour les utilisateurs authentifiés	13
4.1.3. Liaison avec la base de données	14
4.2. Réalisation de la page d'accueil	14
4.2.1. Affichage des photos	14
4.2.2. Barre des tâches	15
4.2.3. Design du site	15
4.2.4. Fonctionnalités des administrateurs	15
4.2.5. Sécurisation du site	16
4.3. Relation avec la base de données	17
4.3.1. Mise à jour des mots de passe	17
4.3.2. Création d'un nouvel utilisateur	17
4.3.3. Requête pour l'affichage des images	17

5 Réalisation du projet complet sur Raspberry (cf. 18)

1. REGROUPEMENTS

1. Base de données et Site Web	
2. Mise en service du Raspberry	
3. Installation de la base de données	
4. Mise en œuvre du site internet	
5. Lancement du test final	

I. Création du projet par partie

1. PLANIFICATION DU PROJET

1.1. Planification des tâches

Nous avons tout d'abord listé toutes les tâches que nous avons pu déjà prévisualiser le jour de la réception du matériel.

Répartition des tâches :

ID		Responsable	Assistant	Avancement
0	Installation Raspberry :	Clara	Matthieu	90 %.
1	Base de donnée :	Benjamin	Matthieu	0 %.
2	Programmation :	Matthieu	Clara, Benjamin	0 %.
3	Site Web :	Benjamin	Matthieu Clara	0 %.
4	Rapport + Présentation	Matthieu	Clara, Benjamin	0 %.

0. Installation Raspberry

↳ .2 install l'image

1. Base de données

- ↳ .1 définir les bases nécessaires
- ↳ .2 les implémenter
- ↳ .3 Accessible pour l'administrateur de ce système
- ↳ .4 Les rendre accessible au site web

2. Programmation :

- ↳ .1 réaliser le main
- .2 définir les fonctions
 - ↳ .2.1 PrintPhoto
 - ↳ .2.2 AffichageLED
 - ↳ .2.3 MoyMdp
- .3 Phase Test

3 Site Web

- ↳ .1 Realisation de la page de connexion
 - ↳ .1.1 Design et mise en page
 - .1.2 fonctionne avec des utilisateurs authentifiés
 - .1.3 Liaison avec la base de données
- .2 Realisation de la page d'accueil (visualisation photo)
 - ↳

1.2. Répartition des tâches

La répartition des tâches est très importante pour le bon déroulement de celles-ci. Nous avons donc décidé de nous attribuer des rôles selon les parties ou même les tâches. Concepteur, Réalisateur et Testeur, seront les rôles répartis sur le groupe afin d'avoir une cohérence entre le temps de travail réparti de chacun.

Concepteur : Personne qui va gérer la tâche et aider le Réalisateur.

Réalisateur : Personne qui va programmer, désigner et régler les problèmes.

Testeur : Personne qui va tester et effectuer un rapport de situation au Réalisateur.

1	Planification du projet	Benjamin	Matthieu	Clara
1.1	Rapport	R	R	R
↳ 1.1.1	Plan	R	C	C
1.3	Planification des tâches	T	C	R
1.4	Répartition des tâches	R	R	R
1.4	Diagramme de Gantt	R	C	C
1.5	Phase de test du projet	T	T	T
2	Base de données			
2.1	Définir la base de données	C	T	R
↳ 2.1.1	Diagramme entité association	R	T	C
2.2	Optimisation de la base de données	R	T	C
2.3	Exécution de la base de données	C	T	R
2.4	Accessibilité de l'administrateur	C	T	R
2.5	Liaison avec le site web (autonome)	R	T	C / R
2.6	Fonctions pour simplifier le code	C / R	T	T
3	Programmation			
3.1	Réaliser le programme principal	C	C / R	C
3.2	Définir les fonctions	C	C / R	C
↳ 3.2.1	Fonction « PrisePhoto »	C	C / R	C
↳ 3.2.2	Fonction Serveur/Client	T	C / R	T
↳ 3.2.3	Fonction « AllumerLED » (non reçues)	-	C / R	-
↳ 3.2.4	Fonction « Mise a jour de la BDD »	C	C / R	C
3.3	Phase de test	T	T	T
4	Site Web			
4.1	Réalisation de la page de connexion	C	C	R / T
↳ 4.1.1	Design et mise en page	C	C	R
↳ 4.1.2	Fonctionnement des utilisateurs authentifié	T	T	C / R
4.2	Réalisation de la page d'accueil	R	C	R / T
↳ 4.2.1	Affichage des photo	R	T	C / T
↳ 4.2.2	Barre des taches (en haut)	T	C	R
↳ 4.2.3	Design de la page	C	C	R
↳ 4.2.4	Fonctionnalités des administrateurs	C	T	R / T

4.3	Mise à jour des mots de passes	C	T	R / T
4.4	Création d'un nouvel utilisateur	C	T	R / T
4.5	Requête pour l'affichage des images	R	T	C
4.6	Mise en service du Raspberry	C	R	T
↳ 4.6.1	Installation apache2, PHP	C	R	T
↳ 4.6.2	Installation phpMyAdmin, Maria DB	C	R	T
↳ 4.6.3	Mise en lien base de données et site web	C	R	T

1.3. Diagramme de Gantt

Une fois la répartition des tâches faite, il nous suffit d'estimer le temps passé sur les tâches ainsi que leurs dépendances pour avoir une idée de la date de rendu du projet. Voici le Gantt que nous avons pu réaliser.

« Diagramme de Gantt **FINAL** »

2. BASE DE DONNEES

2.1. Définir la base de données

Pour définir la base de données, nous avons commencé par définir toutes les informations utiles pour gérer entièrement les données des utilisateurs, des tests et des images. Nous avons alors listé toutes ces données.

Tests : Nom du test, ainsi que son chemin (path).

Personnels : Nom, prénom, mot de passe.

Images : Nom, date, heure, numéro de caméra.

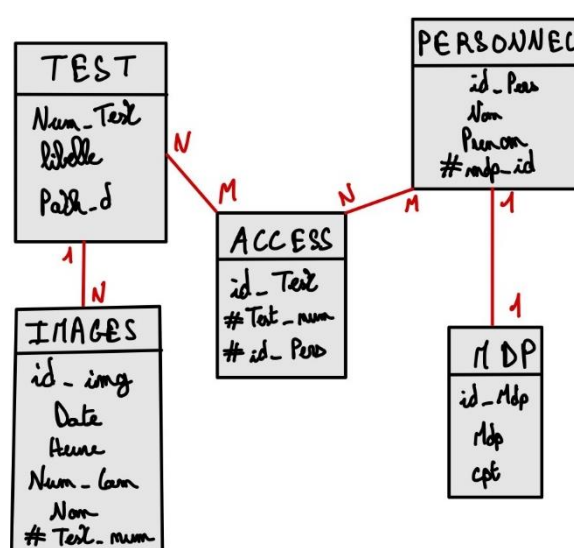
2.1.1. Diagramme entité association

Durant notre première année de formation, nous avons appris à réaliser un diagramme nous permettant d'initialiser une base de données, celui-ci se nomme « Entité / Association ».

Voici notre diagramme Entité / Association finalisé. Cela représente les tables qui seront dans la base de données avec les données de chacune des tables, ainsi que leurs relations.

Les relations sont les traits reliés entre les tables. Ils permettent une certaine liaison entre les tables pour avoir une base de données dynamique et complète.

Le nombre « 1 », « N » et « M » sont transcrits de part et d'autre des relations. Voici des exemples pour avoir une compréhension plus facile de ces significations.



- Exemples :
- De la table Images à Test : Il y a un nombre **N** d'images pour **1** test.
 - De la table Test et Access : Il y a **N** tests avec **M** accès pour accéder aux tests.
 - De la table Access à Personnel : Il y a **N** accès pour plusieurs (**M**) employés.
 - De la table Personnel à Mdp : Il y a **1** employé qui a **1** mot de passe.

De nombreux dièses « # » sont devant certains champs des tables. C'est une manière de décrire que le champ est une clé étrangère. C'est-à-dire qu'elle dépend de la clé primaire de la table voisine. Cela permet une certaine relation entre les tables.

2.2. Optimisation de la base de données

Après plusieurs utilisations de la base de données, des tests ont été fait pour savoir si la base de données était opérationnelle. On s'est aperçu que certaines relations n'étaient pas nécessaires. Nous avons donc ré-établi l'initialisation de la base de données. (*Voir directement le code sql pour la création de la base de données.*)

Nous avons donc conclu de créer 3 tables (PERSONNEL, IMAGES et LOGS). Ces tables suffisent au bon fonctionnement de la base. Les tables contiendront toutes les informations utiles au site web et aux programmes python pour le bon déroulement des tâches à venir.

2.3. Exécution de la base données (Test)

Le code SQL final se trouve dans l'archive SAE24_BMC

(Dossier B.B.B.B)

• Création de la base de données sur PhpMyAdmin (PMA) :

PhpMyAdmin est une application web de gestion de base de données MySQL et MariaDB, réalisée principalement en PHP.

Nos tests se déroulent sur PMA grâce à isis (réseau interne de l'IUT) pour le moment, et nous devons tout simplement installer PMA sur le Raspberry et créer un serveur pour que cela fonctionne de la même manière qu'avec isis. Des tests sont également réalisés sur un de nos ordinateurs en local grâce à easyPHP Deserver, créant une boucle de serveur local en association avec PMA.

Pour la création de la base de données tout se passe depuis l'interface graphique de PMA qui est très intuitive. En étant sur isis, nous avons la possibilité d'utiliser une base de données. C'est pour cela que nous avons directement commencé à créer les tables grâce à ces commandes :

Table PERSONNEL :

```
CREATE TABLE PERSONNEL (
  idUser INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
  nom VARCHAR(70) NOT NULL,
  prenom VARCHAR(50) NOT NULL,
  login VARCHAR(30) NOT NULL,
  mdp VARCHAR(64) NOT NULL,
  admin BOOLEAN,
  cpt_renouvellement INT NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=10;
```

Table IMAGES :

```
CREATE TABLE IMAGES (  
  idImg INT PRIMARY KEY NOT NULL AUTO_INCREMENT,  
  nomImg VARCHAR(70) NOT NULL,  
  nomTest VARCHAR(70) NOT NULL,  
  dateImg DATE NOT NULL,  
  heure TIME NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=10;
```

Table LOGS :

```
CREATE TABLE LOGS (  
  idLog INT PRIMARY KEY NOT NULL AUTO_INCREMENT,  
  action VARCHAR(60) NOT NULL,  
  dateLog DATE NOT NULL,  
  heureLog TIME NOT NULL,  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=10;
```

Les tests pour la création de la base de données ainsi que les tables sont concluants. Nous avons rencontré quelques difficultés pour la création des clés étrangères mais ce fut très vite réglé.

• Ajouter des données dans les tables :

Si les tables sont bien initialisées, alors l'ajout d'enregistrement dans les tables devraient bien se passer. Pour les enregistrements, il est préférable de ne pas mettre des espaces pour éviter toute confusion dans le code, nous en reparlerons plus tard dans la partie du site web. Voici les commandes pour ajouter des enregistrements :

Ajouts des enregistrements :

```
INSERT INTO PERSONNEL (nom, prenom, login, mdp, admin, cpt_renouvellement) VALUES  
( 'faretie','benjamin','fb','fb',TRUE,5),  
( 'perre','matthieu','pm','pm',FALSE,30),  
( 'woodman','clara','wc','wc',FALSE,50);  
  
INSERT INTO IMAGES (nomImg, nomTest, dateImg, heure) VALUES  
( 'Photo1','Test1','2022-04-27','19:08:48'),  
( 'Photo2','Test1','2022-05-27','20:08:48'),  
( 'Photo1','Test2','2022-06-27','21:08:48');  
  
INSERT INTO LOGS (nomLog, dateLog, heureLog) VALUES  
( 'Delete Photo 1','2022-07-27','20:00'),  
( 'Recherche des images du test 2 ','2022-08-27','20:00'),  
( 'Mot de passe modifié du User 1','2022-09-27','20:00');
```

L'ajout d'enregistrement n'est pas compliqué en matière de code mais il faut être judicieux au niveau des contraintes d'enregistrement. Nous avons essayé de manipuler ces enregistrements avec au début des « nomImg » avec des espaces comme « Image 21 » mais l'espace pose des problèmes dans le code php. Nous en concluons qu'il ne faut pas mettre d'espaces dans les champs utilisés dans le code php. Pour supprimer un enregistrement, il faut tout simplement utiliser la commande :


```
DELETE FROM <nom_table> WHERE <nom_champ> = <valeur_champ>
```

2.4. Accessibilité de l'administration

L'accès à la base de données est seulement autorisé aux administrateurs. C'est-à-dire qu'à la création du serveur phpMyAdmin sur le Raspberry, il faut configurer l'identifiant et le mot de passe de l'accès à la base de données. Il sera alors nécessaire de mémoriser les informations de connexion car il y a un risque de perte d'accès au site pour cause d'oubli de mot de passe, l'accès direct à la base de données devient essentiel.

2.5. Liaison avec le site Web (autonome)

Le site web étant codé en HTML/CSS et PHP, nous devons nous connecter depuis le code php à la base de données.

```
try {  
    $bd = new PDO("mysql:host=localhost;dbname=", "id", "mdp");  
    $bd->exec('SET NAMES utf8');  
}catch (Exception $e){  
    die("Erreur: Connexion à la base impossible");}
```

Fig : commande nécessaire à la connexion

Ce petit morceau de code nous permet d'utiliser la base de données définie dans les options de la fonction PDO() (fonction permettant la connexion à la base de données codée en PHP), et si l'accès à la base est refusé, un message d'erreur est retourné indiquant que la connexion à la base de données est impossible.

Pour conclure, la création de la base de données a été un succès et peu de risques ont été découverts, mis à part la perte de mot de passe admin que nous avons déjà anticipé, mais il y a une solution à cela. Il suffirait que l'admin se réfère à l'interface graphique de Phpmyadmin pour ensuite pouvoir retrouver son mot de passe ainsi le modifier. Mais nous en reparlerons dans la partie du site web car le mot de passe sera crypté grâce à la fonction md5() en php. Une fois crypté le mot de passe n'est plus affiché en clair dans la base de données, alors nous définirons une méthode pour ce risque dans la partie Web.

3. PROGRAMMATION

3.1. Réaliser le programme principal

Toutes les fonctions présentées ci-après sont accessibles dans l'archive AE24_BMC

(Dossier Raspberry)

Ce programme permet au Raspberry de prendre des photos et d'agir comme un serveur de requête. Il respecte donc la condition d'être disponible à n'importe quel moment de la journée. Pour cela on automatise le démarrage du fichier. On utilise donc la ressource « Crontab » qui, dans notre cas, lance le script au démarrage du Raspberry (fig 1). Pour permettre ensuite de garder le programme ouvert en permanence, on utilise une boucle sans fin (« While True »). Nous disposons également d'un deuxième programme appelant les mêmes ressources pour faire fonctionner la prise de photo depuis le Web.

```
root@raspberrypi:/home/pi# crontab -l
@reboot python3 /home/pi/Downloads/testmain.py
```

figure 1 : lancement du script de démarrage grâce à Crontab

3.2. Définir les fonctions

Plusieurs fonctions sont nécessaires au bon fonctionnement :

- La 1^{ère} est donc celle qui permet de prendre des photos
- La 2^e permet de démarrer un serveur exploitant la fonction de prise de photo
- La 3^e, elle, teste et reprend une photo en cas de mauvaise luminosité
- Finalement la 4^e nous autorisant à venir communiquer avec le serveur.

Au cours du développement de nouveaux besoins sont apparus :

- La prise de photo depuis le site internet
- La mise à jour de la base de données pour répertorier les images
- La mise à jour de la date de validité des mots de passe

3.2.1. Fonction « PrisePhoto »

Cette fonction est assez simple à mettre en place, il nous suffit de récupérer 2 variables :

- le nom du projet,
- le nom de la photo (non nécessaire selon la provenance de la requête)

Nous avons également besoin de 3 librairies et d'un paquet :

- Le programme FS Webcam
- La librairie os
- La librairie datetime

Pour cela, on utilise le module « FS webcam » qui permet de prendre des photos grâce à des lignes de commandes. Nous devons également pouvoir prendre des photos avec un nombre de caméras variable, ici on suppose que le maximum de caméras connectées est de 4 (4 ports USB sur le Raspberry). Nous avons également besoin de certaines librairies Python : « os » permettant l'envoi de commande depuis le code python au terminal et « datetime » permettant la création du nom de l'image.

Voie d'amélioration : Créer une boucle au lieu d'un test de condition pour permettre un nombre illimité de caméras connectées et permettre une meilleure lisibilité du code.

3.2.2. Fonctions « Serveur/Client »

Après avoir permis l'utilisation de la caméra depuis python, maintenant il nous faut la possibilité de le faire depuis une machine distante. C'est alors que cette fonction prend tout son sens, celle-ci permet donc de communiquer entre un client et le Raspberry. Pour cela, il nous suffit de créer un socket et de récupérer les requêtes émises par le client.

Cette fonction est donc double, une coté serveur (écoute du port) et l'autre du côté du client (envoi sur le port).

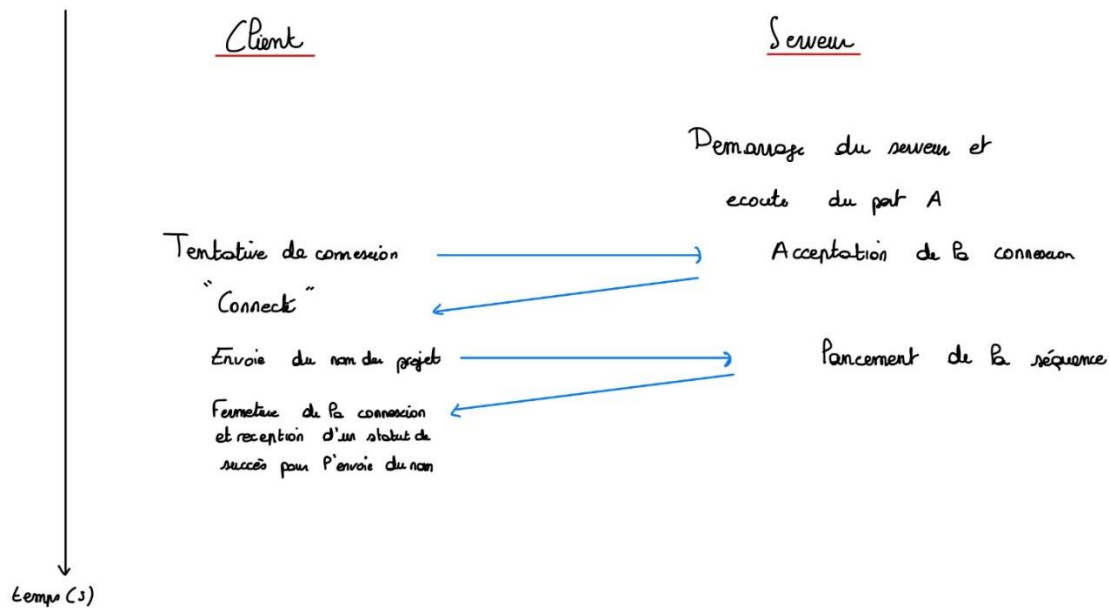


Figure 2 : schéma de la relation client/serveur lors d'une prise de photo

3.2.3. Fonction « AllumerLED »

Cette fonction, comme son nom l'indique, permet l'allumage de la LED et donc la reprise de photos dans de « meilleures » conditions.

Elle se décompose en 2 principales actions :

- Celle de tester le taux de pixels noirs présents sur l'image (50% de pixels noirs sur une image de dimensions 1920*1080px).
- Celle d'allumer la LED et de recapturer les images

A défaut de matériel non fourni, elle n'a pas pu être mise en place mais elle permet quand même de tester le niveau de noir d'une image (non raccordée au programme principal mais le sera avant la présentation).

3.2.4. Mise à jour de la base de données

Maintenant que nous avons pris en photo et stocké sur le Raspberry l'image, il nous faut donc mettre en lien ces images et projets avec le site. On a donc besoin de 2 fonctions :

- La mise à jour dynamique des mots de passe
- L'ajout automatique des images dans la base de données

La 1^{ère} fonction consiste à envoyer une requête et modifier tous les champs de « cpt_renov » (compteur de 90 jours qui permet de vérifier si le mot de passe est encore valable) en enlevant 1 à ces enregistrements par l'appel journalier d'un fichier python via Crontab.

La 2nde fonction, elle, permet d'enregistrer les nouvelles images dans la base de données en envoyant une requête contenant le nom du projet, l'heure, la date et le nom de l'image. Cette fonction est appelée à la suite de la prise de photo.

3.3. Prise de photo depuis le site Web

Cette fonction permet au site web de communiquer avec les fonctions python citées précédemment à l'image de la fonction Serveur. Ici, le site Web envoie une commande dans le terminal « python3 /home/pi/Downloads/PrisePhotoWeb.py projet » (sujet à évolution) ce qui lance le script qui va donc démarrer la prise de photo par l'appel de la fonction « PrisePhoto » en utilisant les variables passées par le terminal. (Nous utilisons la librairie « sys » pour récupérer nos variables)

4. SITE WEB

Maquette du site :

L'entièreté du site est codée en HTML, CSS et PHP.

○ Partie commune à tout utilisateur

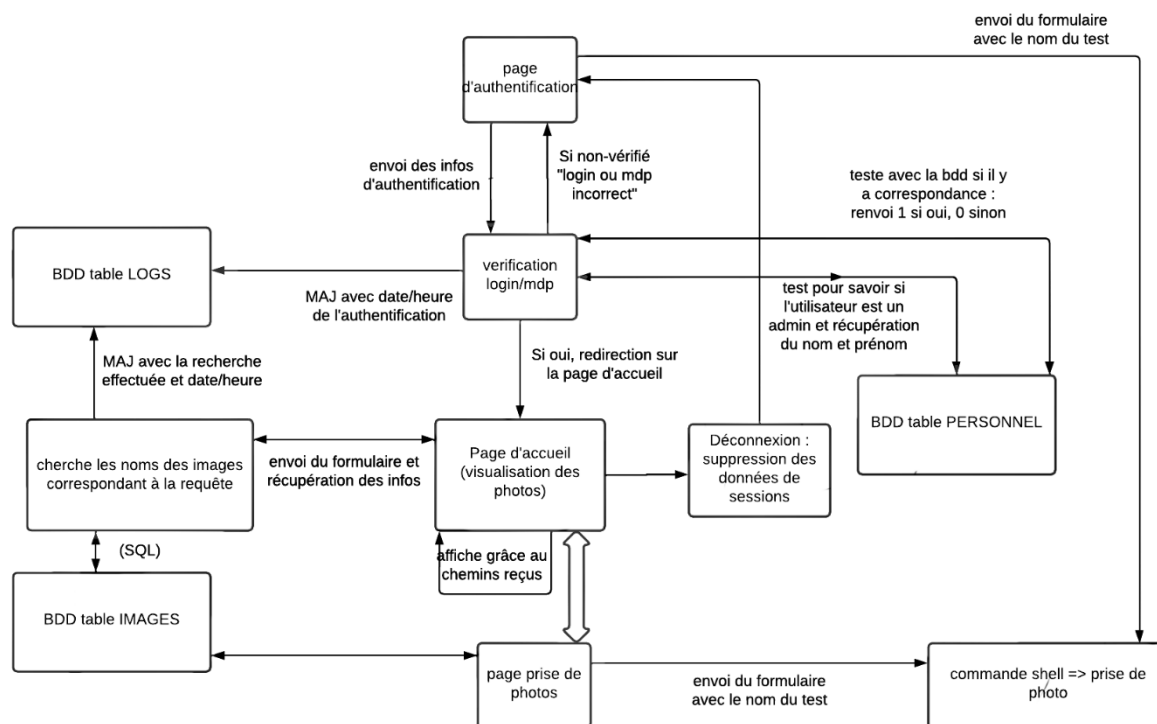


Figure 1 : schéma de fonctionnement du site commun à tout utilisateur

○ Partie administrateur (qui s'ajoute à la partie commune)

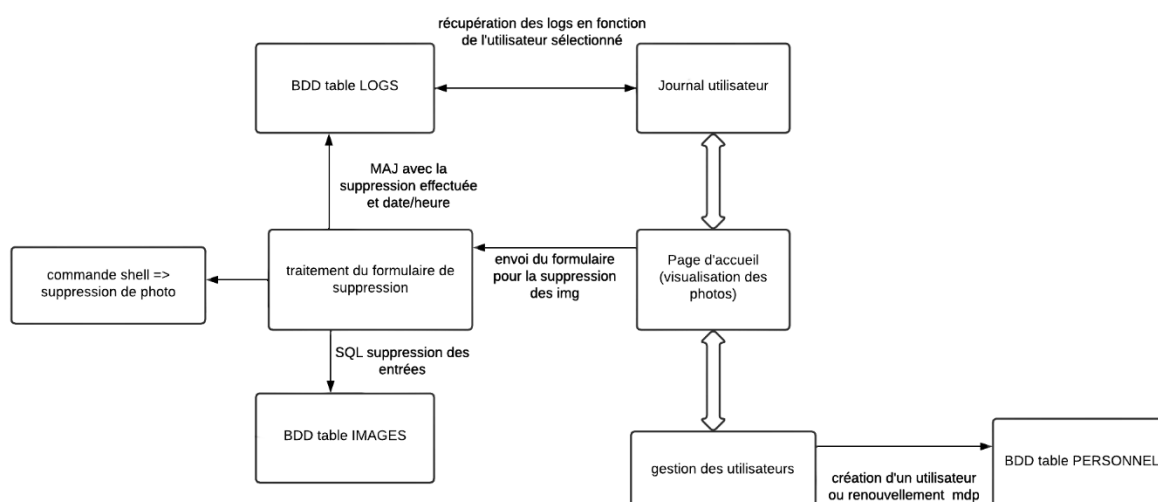


Figure 2 : schéma de fonctionnement de la partie administrateur

4.1. Réalisation de la page de connexion

Dès l'ouverture du site Web via l'adresse A ou 127.0.0, la page de connexion apparaît pour nous permettre d'avoir une certaine restriction selon l'utilisateur qui utilise l'interface du site. Ainsi cela rend la connexion nécessaire à l'accès du reste du site Web.

Par soucis de sécurité, un morceau de code PHP efface toutes les données d'une éventuelle session mal-déconnectée.

4.1.1. Design et mise en page

Pour le design nous avons opté sur un fond d'une couleur similaire au logo Thales pour avoir une unicité des couleurs. Nous avons placé au premier plan au centre de la page le bloc de connexion où il faut renseigner l'identifiant et le mot de passe de l'utilisateur. En cas d'oubli du mot de passe, il faudra systématiquement envoyer un mail à l'administrateur.

Grâce à un système de redirection présent sur toutes les pages et à la méthode GET :

- Un message en rouge se rajoute en cas de :
 - o Mauvais mot de passe ou identifiant
 - o Tentative d'accès à des pages auxquelles un utilisateur n'a pas les droits
 - o Non-connexion
 - o Mot de passe expiré pour un utilisateur non administrateur
- Un message en vert apparaît quand un utilisateur se sera correctement déconnecté.

Un autre bloc permet de prendre des photos depuis le site web. Grâce à une requête SQL, les noms des différents tests existants sont récupérés et affichés dans un menu déroulant mais il y a la possibilité de taper directement le nom du test. Nous avons choisi de proposer les noms des tests existants pour éviter les erreurs de frappe et les doublons.

4.1.2. Fonctionnement des utilisateurs authentifié

Les utilisateurs authentifiés sont ceux qui ont des identifiant et mot de passe pour se connecter au site, c'est-à-dire qu'ils sont enregistrés dans la base de données. Il sera donc possible de visualiser les photos par la suite, en étant authentifié. Ainsi il n'est pas nécessaire d'être authentifié pour prendre une photo car la fonction est disponible sur la page de connexion.

Fonctionnement de l'authentification :

L'utilisateur rentre ses identifiants dans le formulaire et les envoie vers une page PHP qui va :

- Crypter le mot de passe avec la clé de cryptage
- Faire une requête SQL pour vérifier si la combinaison existe
- Récupérer des informations telles que le nom, le prénom, vérifier si l'utilisateur est un administrateur ou non et les stocker dans des variables de session
- Enregistrer l'heure et la date de l'authentification sur la base de données

- Rediriger vers la page d'accueil du site

Exigences accomplies :

Reference PHOTO_ATB-Securite-100

L'utilisation de l'interface de l'application PHOTO_ATB doit être sécurisé par un accès de type login/password.

Reference PHOTO_ATB-Securite-110

Le déclenchement d'une prise de photo par l'application PHOTO_ATB ne doit pas nécessiter l'utilisation d'un compte / mot de passe.

4.1.3. Liaison avec la base de données

La connexion de la base de données depuis le site web est nécessaire au bon fonctionnement de notre site web. Nous nous sommes mis dans la même situation que le Raspberry, en local. Alors l'accès à la base de données se fait tout simplement avec un petit script de 2-3 lignes pour permettre d'utiliser les tables de la base de données. Ce code permet aussi d'afficher s'il y a une erreur avec la base de données.

4.2. Réalisation des pages authentifiées du site

Le site peut se présenter sous deux façons :

- La partie utilisateur normal (ou commune) avec :
 - o La page d'accueil où l'on peut rechercher des images de tests
 - o Une page pour photographier avec un bloc identique à celui de la page de connexion
 - o Un onglet de déconnexion
- La partie admirateur qui présente en plus des pages précédentes :
 - o Une possibilité de suppression des images à l'aide de cases à cocher et d'un formulaire sur la page d'accueil
 - o Une page pouvant afficher les actions effectuées par les utilisateurs (connexions, recherches, suppressions)
 - o Une autre page pour la gestion des utilisateurs sur laquelle les logins des utilisateurs dont le mot de passe doit être changé apparaissent sur la gauche, et au milieu un formulaire dynamique qui peut soit renouveler les mots de passe, soit créer un utilisateur

Exigences accomplies :

Reference PHOTO_ATB-Securite-130

L'application PHOTO_ATB doit limiter l'accès aux différentes fonctionnalités en fonction du profil concerné :

- Administrateur,
- Operateur.

4.2.1. Affichage des photos

Lorsque l'on arrive sur la page d'accueil, seul le formulaire est visible. Une fois qu'une recherche est effectuée, le nom du test apparaît en haut de la page et les images s'affichent dans un tableau en miniature. Il est possible de cliquer sur les photos pour les afficher en plus gros et avoir des informations supplémentaires, telles que la date et l'heure de la prise.

Les actions de suppression et de recherche sont enregistrées dans la table LOGS de la base de données avec l'heure, la date et le login de l'utilisateur.

Exigences accomplies :

Reference PHOTO_ATB-Securite-150

L'application PHOTO_ATB doit mémoriser les actions des différents utilisateurs, avec au minimum :

- La date
- Le user/profil connecté,
- La description de l'action effectuée.

4.2.2. Barre des taches

La barre des taches est l'outil de navigation du site. Elle rappelle les informations de la personne connectée sur la droite, et présente les onglets disponibles selon les droits attribués.



Figure 3 : barre des tâches utilisateur



Figure 4 : barre des tâches administrateur

Nous avons également choisi de notifier le besoin de renouvellement de mots de passe par une police rouge et une bulle de notification. En effet le site étant en local et ne devant pas communiquer avec des appareils extérieur, l'envoi de mail est impossible. Si le nombre de personne ayant besoin d'un renouvellement de mot de passe est nul, alors « gestion des utilisateurs » sera en noir.

4.2.3. Design du site

Le site est sobre et les couleurs respectent la charte graphique de Thalès. L'interface est facile d'accès et permet une navigation simple.

4.2.4. Fonctionnalités des administrateurs

Comme indiqué dans les schémas en fig.1 et 2, les administrateurs ont accès :

- Aux journaux des différents utilisateurs (en cliquant sur leur identifiant dans un bloc sur la gauche)
 - Les actions sont présentées des plus récentes aux plus anciennes
 - Les actions stockées sont de 3 types : connexion, recherche, suppression
 - Les actions sont datées
- A la suppression des photos sur la page d'accueil
- A la gestion des utilisateurs sur une page dynamique avec :

- Sur la gauche les éventuels mots de passe à changer (le bloc n'apparaît pas si tous les mots de pas sont en règle)
- Un formulaire pour le renouvellement de mot de passe
- Un formulaire de création de compte

Fonctionnalités de la page « gestion des utilisateurs » :

- Si le nouveau mot de passe ne correspond pas au format demandé (plus de 8 caractères, une minuscule, une majuscule, un caractère spécial, un chiffre) l'administrateur ne peut pas valider le mot de passe
- Si les deux champs de mots de passe sont différents, un message d'erreur apparaît également
- Les mots de passe sont chiffrés avant d'être envoyés sur la base de données
- Si l'identifiant choisi n'existe pas pour le renouvellement de mot de passe, un message d'erreur s'affiche

4.2.5. Sécurisation du site

Sur chaque début de page, l'authentification est vérifiée : si un utilisateur n'est pas connecté, il est directement redirigé vers la page d'accueil.

```
/*verification de l'authentification*/  
/*redirection vers la page de connexion*/  
session_start();  
if (empty($_SESSION)) {  
    header('Location:../loginpage.php?msg=nonco');  
    exit();  
}
```

Figure 5 : redirection si la session est vide avec le message d'erreur 'nonco'

Sur chaque début de page administrateur il y a aussi une vérification du type d'utilisateur, qui redirige vers la page de connexion avec un message relatif aux droits

```
if (!isset($_SESSION['admin'])) {  
    $_SESSION=array();  
    if (ini_get("session.use_cookies")) {  
        $params = session_get_cookie_params();  
        setcookie(session_name(), '', time() - 42000,  
            $params["path"], $params["domain"],  
            $params["secure"], $params["httponly"]);  
    }  
    session_destroy();  
    if (empty($_SESSION)) {  
        header('Location:../loginpage.php?msg=droits');  
        exit();  
    }  
}
```

Figure 6 : redirection si la session administrateur n'est pas vérifiée avec le message d'erreur 'droits'

Et enfin si l'utilisateur se retrouve sur une page de traitement de formulaire (qui doivent rester 'invisibles' aux utilisateurs), il est redirigé vers la page source du formulaire.

```
/*redirige si l'utilisateur est connecté mais n'a pas envoyé de requête post */  
if (empty($_POST)) {  
    header('Location:../userpages/gestion.php');  
    exit();  
}
```

Figure 7 : exemple de redirection vers la page source du formulaire

4.3. Relation avec la base de données

4.3.1. Mise à jour des mots de passe

Une requête SQL UPDATE, prenant en paramètre l'identifiant de l'utilisateur et le nouveau mot de passe chiffré, met à jour la table PERSONNEL de la base de données. Cette requête réinitialise également le compteur de validité du mot de passe.

```
$sql="UPDATE PERSONNEL SET mdp='$mdp', cpt_renouvellement=90 WHERE login=  
'".$_POST['USER']."'";
```

Figure 8 : exemple de requête SQL pour la mise à jour de mot de passe

4.3.2. Création d'un nouvel utilisateur

Une requête SQL INSERT INTO, prenant en paramètre l'identifiant de l'utilisateur, le nom, le prénom, le statut administrateur ou non et le nouveau mot de passe chiffré, crée un nouvel enregistrement dans la table PERSONNEL de la base de données.

```
$sql="INSERT INTO PERSONNEL (nom, prenom, login, mdp, admin,  
cpt_renouvellement) VALUES ('".$_POST['nom']."', '".$_POST['prenom']."', '".$_POST['USER']."', '$mdp', FALSE, 90);";
```

Figure 9 : exemple de requête SQL pour la création d'un utilisateur

4.3.3. Requête pour l'affichage des images

L'utilisateur rentre le nom du test souhaité et une éventuelle date (partie du site non terminée pour la date mais sera prête pour la soutenance).

Un autre fichier PHP traite le formulaire et récupère dans un tableau de session, le nom des images correspondant à la requête et à afficher, et dans une autre le nom du test essentiels pour choisir le bon répertoire de photos. Nous avons choisi de les stocker dans des variables de session de manière à ce que si un utilisateur qui la page d'accueil et y revient après, les images de la dernière recherche resteront affichées. Une nouvelle recherche écrasera les données de recherches précédentes avec les nouvelles.

Sur la page d'accueil, si les deux variables de session existent, une boucle récupère les images en fonction de la correspondance entre leur nom et le nom dans le tableau.

```
/*galerie d'images*/
$tableau = array();
$dossier = opendir('Images/'.$_SESSION['pathTest']);
/*récupération des images*/
while ($fichier = readdir($dossier)) {
    if (array_search($fichier,$_SESSION['pathImg'])||array_search($fichier,$_SESSION['pathImg'])===0) {
        $tableau[] = $fichier;
    }
}
closedir($dossier);
```

Figure 10 : récupération du nom des images dans un tableau sur accueil.php

4.4. Mise en service du Raspberry

Maintenant que tout fonctionne, il faut mettre en place le site sur le Raspberry, nous avons donc besoin de différent paquets pour gérer la partie Web :

- Apache2 (Service Web),
- PHP (Non installé de base dans le paquet Apache2)




et ensuite des paquets de base de données :









- PhpMyAdmin (gestionnaire de base de données avec interface graphique)
- Maria DB (Base de données)


Notre Raspberry disposant maintenant de tout ces paquets, on se rapproche donc de la phase de test globale.

II. Réalisation du projet complet sur Raspberry

2. REGROUPEMENTS

Historique / Procédure des Test				
Nombre de Tests : 5			Validation finale :	
Phases requissent :		Auteur : Benjamin Membres : Matthieu Clara		 
<ul style="list-style-type: none"> • Base de données fonctionnelle • Site Web Accessible sur le Raspberry • Raspberry opérationnel • Fonctions du Raspberry initialisé 				
N°	Description	Testeurs	État	Remarques
1	Est-ce que la base de données est adaptée et fonctionnelle au Site Web ?	Clara Benjamin		Le Site Web n'a pas de difficulté à s'adapter avec la base de données. Ainsi le site et la base de données est opérationnelle.

2	Implémentation de la base de données sur le Raspberry. Préalable : Installation du serveur phpMyAdmin. (validé)	Matthieu Benjamin		Aucun soucis, en passant par des requêtes SQL.
3	Implémentation du site Web sur le Raspberry. Préalable : Installation du serveur apache2 et de php. (validé)	Matthieu Clara		L'implémentation est un succès, le css est à revoir mais rien de grave.
4	Essayer le site internet depuis le Raspberry. Se connecter, visualiser des photos déjà retranscrites dans la bdd, créer un utilisateur, changer un mot de passe	Matthieu Clara Benjamin		Tout comme en local, sur le Raspberry tout fonctionne avec bien sur toutes les configurations à refaire dans le code. Par exemple : - Ecrire les bons chemins (path) des images.
5	Prendre une photo depuis le site Web (1) Préalable : Créer le bouton pour prendre la photo. (validé)	Matthieu Clara Benjamin		Problème de synchronisation du code python et le code php. Transfère de variables non prises en compte. - Solution 1 : Trouver un moyen d'envoyer des variables du Site Web et les récupérer pour les utiliser dans le code python. - Solution 2 : Écrire du code python depuis le site web.
6	Prendre une photo depuis le site Web (2) « Solution 1 »	Matthieu Clara Benjamin		Le programme python peut récupérer les variables depuis le terminal, il faut essayer d'en envoyer une depuis le site web. ➔ Sera prêt pour la présentation
7	Prendre une photo depuis un client distant (programme 1 caméra)	Matthieu Benjamin		Le poste peut se connecter au Raspberry. La prise de photo depuis un poste distant fonctionne parfaitement bien pour une caméra (ancien programme)
8	Prendre une photo depuis un client distant (programme plusieurs caméra)	Matthieu Benjamin		Le programme ne rentre pas dans les conditions if pour prendre les photos. ➔ Sera prêt pour la présentation
9	Mise a jour de la base de données	Matthieu Benjamin		La fonction de mise a jour des données fonctionne, il faut adapter la connexion à celle déclarée sur le Raspberry. ➔ Sera prêt pour la présentation

10	Mise à jour de la durée de vie des mots de passe	Matthieu Benjamin		<p>Le compteur décrémente comme il faut. Il ne reste qu'à automatiser la tâche pour la rendre autonome et modifier les identifiants de connexion pour permettre la communication avec la base de données du Raspberry.</p> <p>➔ Sera prêt pour la présentation</p>
Conclusion				
<p>Le projet est presque terminé, il ne manque plus que quelques ajustements sur le raspberry dus à notre méthode de division des tâches. Le rendu final sera perfectionné et entièrement testé.</p>				