|  |  |
| --- | --- |
|  | SAE23 + SAE24  Expérimental XTRA Photo (EXP) (PHOTO\_ATB) |

Text

Description automatically generated

|  |  |
| --- | --- |
| N° Projet | BUTRT-2022-1 |
| Nom du groupe | Les Heckers |
| Direction émettrice (nom du sponsor) | Thales Alenia Space |
| Responsable MOA | Philippe CAM |
| Responsable MOE | Jean-Baptiste Bruneau |
| Description synthétique | Développement d’un équipement afin de réaliser une prise de photo d’un banc avionique |

RApport de projet

|  |  |  |
| --- | --- | --- |
| Rédaction – participants à la co-construction du rapport | | |
| Partie prenante | Prénom Nom | Rédacteur (R) / Participant (P) |
| MOA (Project Owner) | Philippe Cam | P |
| MOE (Chef d’équipe) / Test manager | Jean-Baptiste Bruneau | R |
| Référent Technique | Maël Albany | R |
| Développeur | Nicolas Brimboeuf | R |
| Développeur | Selyan Abdallah | R |

Sommaire du rapport

[Documents applicables 4](#_Toc105492076)

[1. Rappel de la problématique 5](#_Toc105492077)

[1.1 Contexte 5](#_Toc105492078)

[1.2 Rappel des objectifs du projet 5](#_Toc105492079)

[1.2.1 Objectifs principaux : 5](#_Toc105492080)

[1.2.2 Objectifs secondaires : 5](#_Toc105492081)

[1.3 Estimation des gains 5](#_Toc105492082)

[1.3.1 Gains qualitatifs liés à la mise en place de l’automatisation 5](#_Toc105492083)

[1.4 Architecture générale de la solution 6](#_Toc105492084)

[1.4.1 Description des matériels 6](#_Toc105492085)

[1.4.2 Description des choix logiciels 6](#_Toc105492086)

[1.4.3 Versions testées 6](#_Toc105492087)

[1.4.4 Composants nouveaux apportés par la solution 7](#_Toc105492088)

[1.4.5 Logiciels utilisés pour construire la solution 7](#_Toc105492089)

[2. Délais 8](#_Toc105492090)

[2.1 Jalons du projet 8](#_Toc105492091)

[2.2 Macro-planning 8](#_Toc105492092)

[3. Risques 10](#_Toc105492093)

[4. Gestion du projet 11](#_Toc105492094)

[4.1 Organisation, répartition et planification des taches 11](#_Toc105492095)

[4.2 Taches Effectuées et taches non effectuées 11](#_Toc105492096)

[4.2.1 Au 14/01/2022 11](#_Toc105492097)

[4.2.2 Au 20/05/2022 11](#_Toc105492098)

[4.3 Problèmes rencontrés 12](#_Toc105492099)

[4.3.1 Au premier semestre 12](#_Toc105492100)

[4.3.2 Au second semestre 12](#_Toc105492101)

[4.4 Retour d’expérience personnelles 23](#_Toc105492102)

[4.4.1 Retour de Jean-Baptiste 23](#_Toc105492103)

[4.4.2 Retour de Mael 23](#_Toc105492104)

[4.4.3 Retour de Nicolas 23](#_Toc105492105)

[4.4.4 Retour global du groupe 23](#_Toc105492106)

[5. Solution proposée 13](#_Toc105492107)

[5.1 Notre organisation 13](#_Toc105492108)

[5.2 Le manuel d’installation et d’utilisation de la solution 14](#_Toc105492109)

[5.3 Fonctions de la solution 14](#_Toc105492110)

[5.4 Choix effectués 15](#_Toc105492111)

[5.4.1 Prise de photos 15](#_Toc105492112)

[5.4.2 Installation automatique 15](#_Toc105492113)

[5.4.3 BDD 15](#_Toc105492114)

[5.5 Explication d’un extrait du code 17](#_Toc105492115)

[5.5.1 Extrait du P3 17](#_Toc105492116)

[5.5.2 Extrait du SW 19](#_Toc105492117)

|  |  |  |
| --- | --- | --- |
| Historique des versions | | |
| Date | Version | Commentaire |
| 07/05/2022 | 1.0 | Première version |
| 20/05/2022 | 4.8 | Version finale avant le démarrage des TEST |
| 07/06/2022 | 5.9 | Version finale |

## Documents applicables

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Titre | Version | Date |
| Excel | Gant v4.xlsx | 4.3 | 14/04/2022 |
| Word | Plan\_de\_validation.docx | 2.0 | 02/06/2022 |
| Word | Doc\_Manuel.docx | 2.3 | 07/06/2022 |
| Fichier texte | Readme.md | 1.0 | 05/06/2022 |

# Rappel de la problématique

## Contexte

Dans le cadre d’un projet de classe pour l’entreprise THALES ALIENA SPACE, nous devons programmer une prise de photo d’un banc avionique avant chaque exécution de test ou sur demande.

Ce projet à débuter le 28 octobre 2021, et nous devons rendre la solution documentée le 16 juin 2022.

## Rappel des objectifs du projet

### Objectifs principaux :

* Avoir un système de prise de photos à intervalle régulier avec la possibilité de changer cet intervalle de temps entre deux prises.
* Prendre des photos sur demande.
* Visualiser les photos sur le site web.

### Objectifs secondaires :

* Rajouter des informations à un groupe de photos (description du test et/ou des changements effectués lors du test, …).
* Sécurisation du site web.

## Estimation des gains

### Gains qualitatifs liés à la mise en place de l’automatisation

Ce projet permet au client de garder une trace de la configuration mise en place lors des tests effectués sur le banc avionique.

Nous prévoyons que la mise en œuvre des applications livrées par ce projet permettra au client de gagner du temps et de la traçabilité lors des tests qu’il réalisera.

## Architecture générale de la solution

### Description des matériels

La MOA nous a demandé d’utiliser les composants matériels suivants :



* Raspberry-Pi 3 modèle B
* A close-up of a microchip

  Description automatically generated with medium confidenceModule caméra UVS 5MP OV5648 Autofocus USB

### Description des choix logiciels

Nous avons choisi d’utiliser différents logiciels open-sources intégrés dans notre solution. Ces logiciels sont les suivants :

* Rasbian (<https://www.raspbian.org/>), système d’exploitation choisi pour les raspberry-pi
* Apache2 (<https://httpd.apache.org/>)
* Php-server et son module php-mysql (<https://www.php.net/>)
* Maria-DB (<https://mariadb.org/>)
* Python3, dont les modules suivants installés avec pip :
  + mysql-connector-python (<https://pypi.org/project/mysql-connector-python/>)
  + python-crontab (<https://pypi.org/project/python-crontab/>)
* Cron (<https://wiki.debian.org/cron>)
* Fswebcam (<https://github.com/fsphil/fswebcam>)

### Versions testées

Notre solution ayant était créé avec des dépendances, nous avons testé le bon fonctionnement de notre programme uniquement avec les versions des dépendances suivantes. Nous ne pouvons donc garantir le bon fonctionnement de notre solution si d’autres versions sont utilisées.

* Apache2 : v.2.4.53
* Php : v.8.1.6
* Fswebcam : v.20140113-1
* Python3 : v.3.9 et v.3.10
* Cron : v.3.0pl1-137
* Maria-DB : v.10.8.3

### Composants nouveaux apportés par la solution

Notre solution intègre les modules suivants :

* P3 : Programme de prise de photo
* SW : Site WEB
* BDD : Base de données
* PUI : Programme d'upload des images
* SS : Sécurité du site

|  |
| --- |
|  |

### Logiciels utilisés pour construire la solution

* IDE PyCharm pour la programmation en python.
  + <https://www.jetbrains.com/pycharm/>
* Visual Studio Code pour la programmation en HTML et PHP.
  + <https://code.visualstudio.com/>
* phpMyAdmin pour l’administration de la base de données.
  + <https://www.phpmyadmin.net/>
* GitHub pour le versionning et la gestion des problèmes :
  + <https://github.com/Swinir/SAE-24_Experimental_XTRA_Photo>

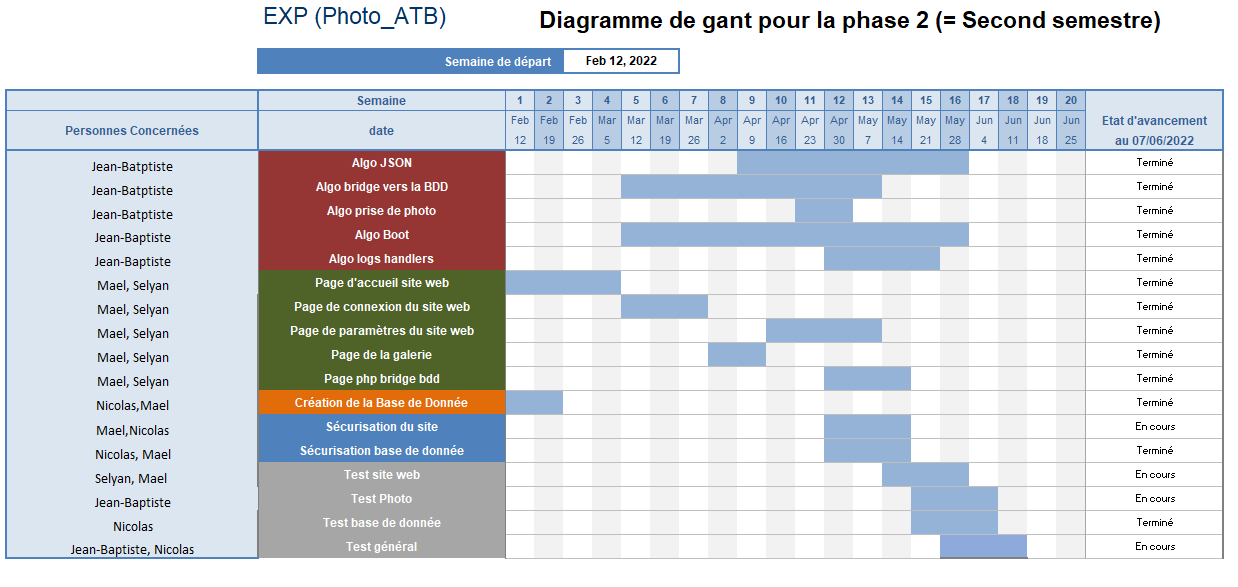
# Délais

## Jalons du projet

|  |  |  |
| --- | --- | --- |
| Phase du projet | Début | Fin |
| Cadrage | 18/10/2021 | 19/11/2021 |
| T0 du projet (première réunion du projet) | 19/11/2021 | |
| Elaboration | 19/11/2021 | 14/01/2022 |
| Construction | 14/01/2022 | 30/05/2022 |
| Test et validation | 15/05/2022 | 15/06/2022 |
| Clôture | 16/06/2022 | |

## Macro-planning





# Risques

|  |  |  |  |
| --- | --- | --- | --- |
| Description du risque | **Probabilité**  1 à 4  Faible à Très fort | **Impact**  1 à 4  Faible à Très fort | Action de couverture de risque |
| Luminosité faible la nuit à cause de manque d’éclairage de la pièce | 4 | 3 | Mise en place de Flash pour avoir une source de lumière pour le bon fonctionnement des caméras |
| Manque de visibilité sur tous les câbles branchés sur le banc | 1 | 4 | Géré par la MOA |
| Maintenance de l’application à la suite de la clôture du projet | 4 | 3 | Documentation |
| Maitrise du planning projet | 2 | 3 | Point hebdomadaire de suivi du planning Gantt |

# Gestion du projet

Dans ce chapitre, nous présentons notre expérience sur l’organisation, la répartition et la planification des taches. Nous donnons également notre retour d'expérience avec les aspects positifs et négatifs, et ce qu'il faudrait faire pour les améliorer.

## Organisation, répartition et planification des taches

Nous avons choisi de répartir les tâches entre les différentes personnes du groupe. Chaque semaine, nous dressions une liste de taches à réaliser pour chaque personne. Néanmoins, vu la complexité de certaines tâches, il n’était pas rare que 2 personnes coopèrent sur la même tâche. Chaque semaine, nous faisions un point pour partager l’avancement de chaque tache et pour remettre en commun le travail réalisé.

Cette organisation nous a permis d’avancer sur plusieurs fronts à la fois. Même si elle nécessite plus de coordination et demande une attention toute particulière à l’assignation des tâches (pour ne pas surcharger un membre de l’équipe par exemple), elle nous a permis de gagner en efficacité.

Nos taches étant planifiées dans le Gantt (section 3.2), il était simple de les suivre pour vérifier si nous étions en retard ou en avance et réajuster le planning ou l’organisation de l’équipe en fonction.

## Taches Effectuées et taches non effectuées

### Au 14/01/2022

Nous avons pu effectuer toutes les taches que nous voulions effectuer pour le 14/01/2022. Cependant, il nous reste tout de même à finir les algorithmes et les supports de présentation pour pouvoir considérer toutes les tâches prévues pour le premier semestre terminées. Nous aurions voulu finir tous les algorithmes avant de commencer à planifier le déroulement de la phase de construction des applications, cela nous aurait permis de mieux gauger le temps a alloué pour chacune de ces taches.

### Au 20/05/2022

Durant cette seconde partie nous avons pu finir de coder les algorithmes, le site web et la base de données. Il nous reste à faire les tests de chaque MOE puis les tests finaux avant la restitution du produit au client. Nous sommes dans les temps, nous respectons nos exigences et le Gantt. De plus certains objectifs secondaires comme la photo prise par impulsion manuelle sur bouton physique par exemple, ont été annulé après discussion avec le client.

### Au 07/06/2022

Nous avons pu réaliser toutes fonctions que nous voulions en priorité, il nous reste seulement quelques fonctions secondaires à finaliser avant notre présentation.

## Problèmes rencontrés

### Au premier semestre

Au premier semestre nous n’avons pas rencontré de problèmes techniques. Nos difficultés concernaient principalement la compréhension du sujet. Pour certains d’entre nous c’était notre premier gros projet de groupe, il a donc fallu apprendre à nous connaitre afin de repartir plus efficacement les tâches grâce au Gantt.

Il a fallu aussi qu’on trouve du temps en commun en plus de ce qu’il nous a été fournis pour faire des récapitulatifs. Ceci nous a appris que c’est compliqué de trouver du temps libre au même moment pour plusieurs personnes. Les problèmes d’organisation sont plus compliqués à gérer que ce que l’on pensait.

Tout de même nous sommes fières de ce que l’on a réalisé dans cette première phase du projet qui nous a bien aidé et facilité la tâche sur l’étape de réalisation du second semestre.

### Au second semestre

Nous avons rencontré plusieurs problèmes importants au second semestre. Nous avons choisi d’utiliser GitHub pour les référencer sous forme de problèmes (issues) afin d’en garder l’historique :

<https://github.com/Swinir/SAE-24_Experimental_XTRA_Photo/issues?q=is%3Aissue+is%3Aclosed>

Cependant, cette page ne contient pas une liste exhaustive de toutes les difficultés que nous avons rencontrées. En effet nous avons résolu de nombreuses difficultés directement en codant l’application.

Néanmoins, nous voudrions insister sur certaines difficultés :

* La gestion de plusieurs caméras a créé des problèmes plus complexes que ce nous avions anticipé.
* La synchronisation des logs entre les modules P3 et SW nous a pris beaucoup plus de temps que ce que nous avions prévu.
* Lors de la mise en place de la prise de photo automatique, nous avons rencontré des problèmes de lecture de fichiers. Cela est dû à l’exécution des commandes dans le Crontab et le PHP. Le Crontab exécute les scripts au niveau du dossier utilisateur ce qui empêche les scripts python de trouver les autres fichiers. Cela a été résolu avec l’ajout d’un cd avant l’exécution du script python. Pour le PHP, il a fallu que nous changions l’utilisateur avec lequel le processus apache2 est exécuté car l’utilisateur par défaut ne possède pas les droits d’ouverture des flux vidéo.

# Solution proposée

## Notre organisation

A screenshot of a phone

Description automatically generated with low confidenceNous avons choisi de diviser le projet en plusieurs parties (P3, SW, BDD, …). Nous avons créé chaque module pour qu’il soit indépendant. Vous pouvez les retrouver dans les différentes branches de notre GitHub :

Les branches s’organisent de la façon suivante :

* Des branches wip-XXX sur lesquels les développeurs peuvent directement envoyer des modifications : il est donc possible de commit sur ces branches. Ce sont des branches dites « instables ». En effet, aucune des modifications sur ces branches ne sont vérifiées par les autres membres de l’équipe.
* Des branches main-XXX où les développeurs ne peuvent pas directement envoyer de modifications (donc commit impossibles). Pour envoyer du code dans ces branches, il faut effectuer une demande de récupération de code depuis une autre branche (une branche wip-XXX). Cette demande doit être approuvée par au moins une autre personne pour qu’elle soit acceptée et que le nouveau code soit ajouté. Ce sont donc les branches « stables » des différents modules de notre solution.
* La branche par default, la branche main. C’est la branche depuis laquelle nous créons les versions que nous communiquons au client.

Nous pouvons voir toutes les pull request que nous avons effectuées dans l’onglet :

<https://github.com/Swinir/SAE-24_Experimental_XTRA_Photo/pulls?q=is%3Apr+is%3Aclosed>

Ces pull request permettent de basculer des modifications de la branche wip à la branche main.

## Le manuel d’installation et d’utilisation de la solution

Nous avons choisi de réaliser ce manuel dans un fichier word à part pour ne pas saturer le rapport.

Vous pouvez le retrouver dans le fichier suivant : *manuel.docx*

Vous pouvez aussi retrouver un manuel d’installation plus concis dans le fichier : *readme.docx*

## Fonctions de la solution

Toutes les fonctions demandées par la MOA ont été réalisés en priorité.

Liste des fonctions réalisées demandées par la MOA :

* Prise de photo automatique toutes les 24 heures. Cette fonction est réalisée en python à l’aide de la commande Cron. Cette prise de photo automatique est activable ou désactivable directement depuis le site web. Le code php exécute un script bash préalablement créé lors de l’installation.
* Gestion de la luminosité des photos. La gestion de la luminosité est gérée par le programme  
  P3. Si la photo est trop sombre, une nouvelle prise de photo est réalisée avec un flash. Le seuil à partir duquel la photo est considérée comme sombre est configurable dans le fichier *luminosite.py*. Cependant, comme demandé par la MOA, nous avons laissé vide la partie pour activer le flash : elle est marquée par des commentaires dans le fichier *luminosite.py.*
* Prise de photo depuis le site web. La prise de photo et l’insertion des données dans la BDD sont gérées par le P3. Cependant, depuis le site web, l’utilisateur a la possibilité de rajouter des informations (description et titre) à la capture. Cette dernière fonctionnalité est réalisée en php avec une communication directe avec la BDD.
* La prise de photo peut être prise par plusieurs appareil en même temps, nous avons mis en place un système pour savoir combien de caméra sont connecté pour savoir le nombre de photo à prendre.
* Le site web est protégé avec un système de log-in, pour accéder aux photos il faut absolument être connecté sur un compte qui a des restrictions au niveau de leur mot de passe. Cependant pour accéder aux données sensibles comme la création/mondification/suppression de compte ou bien des logs il faut être connecté avec un compte admin.
* Un système de logs a été mis en place. Dans celui-ci, toutes les erreurs survenues lors de l’exécution du P3 ou du SW sont stockées dans BDD.

Nous avons pris la liberté d’étendre notre solution et de répondre à des besoins non formulés par la MOA. Ces fonctions nous ont paru importantes pour le bon fonctionnement de notre solution et pour aider à la prise en main de notre logiciel par l’utilisateur.

Fonctions réalisées en complément des demandes de la MOA :

* On sait aussi permit de rajouter des fonctions pour aider le visionnage et la recherche de photo dans la galerie comme déjà pouvoir mettre des photos en favori et les trie en ordre alphabétique ou inverse des nom utilisateur ayant pris une photo, on peut aussi trier par date de la plus ancienne à la plus recette et vice versa
* Installation automatique de la solution sur un Raspberry-Pi. Il suffit à l’utilisateur de télécharger la dernière version du logiciel disponible dans l’onglet « Release » de GitHub, puis de le décompresser dans un répertoire. Ensuite, il faut exécuter une simple ligne de commande pour lancer l’installation complète du logiciel (cf : *readme.txt*)
* Nous sommes actuellement en train d’implémenter un système pour visualiser l’espace de stockage disponible pour les photos dans le Raspberry-Pi, il sera visible uniquement depuis un compte admin.

## Choix effectués

Durant la réalisation, nous avons eu la liberté d’utiliser plusieurs langages (python, php, html, …). Nous avons donc fait le choix de réaliser les fonctions dans les langages de programmation qui nous paraissaient les plus judicieux.

### Prise de photos

Pour la prise de photo, le logiciel devait être codé en Python3, ce que nous avons fait. Nous utilisons l’exécutable linux Fswebcam pour récupérer une image depuis le flux vidéo de la caméra. Notre programme est capable de détecter automatiquement le nombre de caméra connectées et de s’adapter en prenant une photo depuis chaque camera.

### Installation automatique

Pour l’installation automatique, nous avons fait le choix de réaliser un script bash qui permet d’installer automatiquement toutes les dépendances de notre programme. Il nous permet aussi de mettre en place la base de données et de « charger » le fichier BDD.sql. Le script permet également la configuration du site web en copiant les fichiers nécessaires dans les bons répertoires et en changeant la configuration du serveur apache2 pour permettre la bonne communication entre le SW et le P3.

### BDD

Pour la base de données nous avons décidé de la créer à partir de phpMyAdmin et de l’héberger sur MariaDB. Elle est composée de 3 tables :

- la table logs , qui comporte 4 enregistrements :

-id\_log : la clé primaire et l’id du log qui est obligatoire

-contenue\_log : l’enregistrement qui va contenir le texte de l’erreur

-niveau\_log : varie entre 0 et 3 selon le niveau de gravité de l’erreur

-date\_log : la date à la quel le log a été créé

- la table photos, qui comporte 7 enregistrements :

-id\_photo : la clé primaire et l’id de la photo

-#user\_photo : clé étrangère pour relier à la table users. Elle permet de spécifier si un utilisateur est connecté et donc de le mettre comme auteur de la prise de photo. Elle est facultative car on peut prendre un photo même si on n’est pas connecté à un compte.

-description : texte contenant la description de la photo, qui peut être facultative. -name\_photo : contient le nom de la photo, qui peut être facultatif.

-favori : de type tinyint, sa valeur est 0 si la photo n’est pas en favori ou égale à 1 si la photo est en favori.

-date\_photo = contient l’heure à la quel la photo a été prise.

-path\_photo = contient le chemin vers la photo dans le Raspberry-Pi.

- la table users, qui comporte 7 enregistrements :

-id\_user = clé primaire et l’id de l’utilisateur.

-#username = contient le pseudo de l’utilisateur et est la clé étrangère pour relier l’utilisateur à la table logs et photo.

-login = contient le l’identifiant du compte.

-password = text contenant le mot de passe du compte.

-admin = type tinyint si = 1 le compte est administrateur / si = 0 le compte est simple utilisateur.

-block\_user = type tinyint si = 1 le compte est bloqué / si = 0 le compte est libre.

-duration = contient la date pour vérifier depuis combien de temps existe pour répondre un besoin de sécurité.

## Explication d’un extrait du code

### Extrait du P3

La partie du code que nous avons choisi d’expliquer pour le P3 est le fichier *logs\_handler.py*. Ce fichier a pour fonction de gérer tous les logs du P3 et de les insérer (dans certains cas) dans la BDD.

Text

Description automatically generated

Tout d’abord, les 4 premières lignes permettent d’importer les modules et les autres fichiers nécessaire à la bonne exécution du code qui suit. L’importation du module Time nous permet d’obtenir le temps actuel de l’hôte sur lequel le script est exécuté. Ce module est compris par défaut dans l’installation de python, nous ne l’avons donc pas modifié. Le fichier import\_data a été codé par nous-même, il nous permet de lire un fichier json puis de retourner une liste contenant tous les objets d’une classe que ce même fichier contient (ici les objets de la classe logs stockés dans le fichier *LOGS.json*). Le module save\_data est aussi un module que nous avons codé, il a en quelque sorte la même fonction que le module import\_data mais au lieu de lire un fichier et de charger les données, il fait le contraire, il prend en paramètre une liste d’objets d’une classe et la sauvegarde dans la bonne syntaxe dans un fichier json (ici *LOGS.json*). Finalement, nous importons le module standard atexit nous permettant d’exécuter des fonctions au moment de l’arrêt normal de l’interpréteur python.

Après l’importation de ces fichiers, vient la déclaration de la classe qui définira les objets que nous allons ajouter (ou lire depuis le fichier *LOGS.json*). Cette classe contient uniquement un constructeur. Ce constructeur nous permet de créer trois attributs appartenant à l’objet. Le premier attribut contiendra la sévérité du log, le deuxième contiendra la description et finalement, le troisième permettra de sauvegarder l’heure de création du log.

Ensuite, on initialise la liste LOGS\_container, c’est elle qui contiendra les objets de la classe logs. On charge les objets existant dans le fichier *LOGS.json* dans un dictionnaire (LOGS\_container\_list). Ce dictionnaire est temporaire et nous permet uniquement d’insérer plus facilement les données du fichier dans la liste LOGS\_container.

Ce code ne figure pas dans une fonction car il est indispensable qu’il soit appelé avant chaque ajout de log. Ce code sera donc systématiquement exécuté dès la mention de la ligne *import logs\_handler*.

Après avoir lu les anciens logs depuis le fichier, la fonction entry\_create est créé, c’est elle qui va permettre au programme d’ajouter un log. Elle reprend en paramètre les mêmes informations que la classe logs.

La ligne *import sql\_bridge* ne figure pas tout en haut du fichier, cela est fait pour éviter de générer une nouvelle connexion à la base si le programme n’a pas besoin de rajouter un log (donc s’il n’y a pas d’appel de la fonction entry\_create).

Comme nous voulions avoir la possibilité pour l’utilisateur de lire les logs directement sans avoir à déchiffrer le fichier json, nous avons ajouté un fichier *LOGS.txt*. Ce fichier contient les mêmes informations que le fichier json, seulement, il est bien plus lisible avec un logiciel de traitement de texte. Comme le fichier json, le fichier texte doit être lu au préalable pour que l’on puisse ajouter un log sans écraser tout son contenu.

La ligne 25 permet de récupérer l’heure sur la machine qui exécute le script. La ligne 26 permet de mettre en forme la réponse du module Time pour pouvoir l’envoyer dans la BDD. Dans la BDD, le champ date\_log est au format time. Cela veut dire que pour insérer de nouvelles informations, celle-ci doivent respecter le format suivant : YYYY(year : année)-MM(mois)-DD(day : jour) HH(heure):MM(minute):SS(seconde) (cf : <https://dev.mysql.com/doc/refman/8.0/en/datetime.html>).

Ensuite, le programme fait correspondre le niveau de sévérité en chiffre avec son équivalent en texte. Nous aurions voulu réaliser cette partie avec un switch case. Mais cela n’existant pas dans la version 3.9 de python, et ne voulant pas forcer l’utilisateur à utiliser python 3.10 uniquement pour cette fonctionnalité, nous avons adopter une approche plus classique avec des if en séquence.

A la ligne 40, nous mettons en forme le nouveau log. Cette variable « result » va nous être utile uniquement pour écrire dans le fichier texte, ce que nous faisons 2 lignes en dessous avec le « output.write ».

Le test de la variable sql permet de déterminer si le log doit être sauvegarder dans la BDD. Ce test est nécessaire par exemple lors de l’installation automatique de notre programme car des logs sont générés avant que l’on puisse effectuer une insertion de données dans la BDD.

Lorsque la variable sql vaut « yes », nous préparons les données à envoyer dans la base avec la variable « values ». Puis nous ajoutons ce nouveau log à la classe (pour sa future écriture dans le fichier json). Ensuite nous démarrons la connexion avec la base et nous insérons les données.

Si la variable sql vaut « no », nous ajoutons uniquement ce nouveau log à la classe logs.

La fonction exit\_handler\_logs définie ligne 55, nous permet de préparer la commande de sauvegarde de la liste d’objets de la classe logs. Cette sauvegarde s’effectuera uniquement au moment de l’arrêt normal de l’interpréteur python. Cela est rendu possible grâce au module atexit que nous avions préalablement importé.

### Extrait du SW

Pour la partie SW (Site Web), nous avons choisis de parler de la prise de photo. Celle-ci est répartie sur deux pages :

* Le formulaire (HTML/CSS),
* Le script de capture (PHP) ;

Une image contenant texte

Description générée automatiquementCommençons par le « formulaire » :

Pour le formulaire, nous avons créer une page en HTML/CSS. Les lignes 1 à 7 composent ce que l’on appelle l’entête de la page. C’est les informations relatives à la page, comme : le « type » de page (< !DOCTYPE html>), le début de la page et sa langue (<html lang= ‘fr’>) , son encodage (<meta charset=’UTF-8’>), puis il y a le lien vers la feuille de style (CSS) utilisée (<link rel=‘stylesheet’ href=‘style.css’> car le CSS s’appelle « style.css ») et enfin le titre de la page, qui ici est vide.

Ensuite, dans la balise body, se trouve tout ce qui va apparaître sur la page.

Tout d’abord, nous avons fait des inclusions de codes, grâce à la fonction « include() » située dans la balise PHP. Nous avons préféré faire des inclusions car ce sont des parties de la page qui reviennent souvent dans les autres pages du site. En effet, celles-ci correspondent au bandeau supérieur de la page (« header.php ») et au menu dynamique situé à gauche sur la page (« menu.php »). Cela permet :

* De ne pas réécrire plusieurs fois les mêmes lignes de codes dans plusieurs pages différentes,
* Mais aussi, rendre la modification, de ces deux parties, plus simple.

Ensuite, nous avons une balise « section » (appelée « main » (principale) pour simplifier le CSS). Celle-ci permet, à la base, de créer un espace pour séparer la page en plusieurs parties. Mais ici, celle-ci sert juste à structurer la page et permettre l’ajout d’éléments sur cette dernière, si changements il y a.

Dans cette balise section, nous avons mis un bout de code PHP. Ce bout de code permet l’affichage des messages « d’erreurs ». Celui-ci utilise les variables globales $\_GET, présentes dans l’URL de la page. Ici, il s’agit d’informer l’utilisateur sur les actions effectuées sur la page de connexion : soit il est connecté (si la page de connexion renvoie ‘1’), soit son compte est bloqué (si la page renvoie ‘2’).

Pour la présentation de la page, nous souhaitions mettre un titre en gros, donc nous avons utiliser la balise de titre « <h1> ». Elle sert à mettre un texte présenté comme un titre (en gros et en gras).

Ensuite, nous avons fait la partie la plus importante de ce site : le formulaire de capture.

Celui-ci est composé de trois éléments :

* Un champ pour le titre de la capture, (l.29)
* Un champ pour la description de cette capture, (l.33)
* Et un bouton de capture. (l.36)

Tous ces éléments sont placés dans la balise « form ». Cette dernière, si elle en reçoit l’instruction, renvoie vers une page spécifique (ici « photo.php »), en envoyant aussi les variables qu’elle contient, sous la forme d’une variable $\_POST[‘nom du champ dans le formulaire’].

Entre les différentes balises « p » (paragraphes), nous avons les éléments constituants notre formulaire.

Le premier élément est un champ de type « text » qui a pour nom : « nom ». La partie « placeholder » est un texte qui va être en fond si rien n’est écrit dans le champ texte.

Le second élément, lui, est un champ de type « textarea ». Ce type de champ, différent de « text », est destiné à accueillir des textes un peu plus volumineux (idéal pour une description). Les parties « rows » et « cols » définissent la taille originale du champ. On retrouve aussi la même partie « placeholder » que dans le champ précèdent.

Enfin, on trouve en troisième élément, un bouton de type « submit ». C’est lui qui donne l’instruction, précédemment évoqué, au formulaire de s’exécuter et d’envoyer les variables et leur contenu à la page suivante.

Une image contenant texte, intérieur, capture d’écran

Description générée automatiquementLa page suivante est le script en PHP de capture :

Tout d’abord, c’est un script, donc il n’y a pas d’affichage.

Le script commence avec un appel de la fonction « session\_start() ». Cela nous permet d’utiliser les variables globales de session ($\_SESSION). Ensuite, nous avons, encore une fois, une inclusion de code, ici, pour la connexion à la base de données.

Puis, nous récupérons les variables du formulaire de la page précédente :

* $nom : le champ nom de la capture s’il n’est pas vide,
* $descrip : le champ description de la capture s’il n’est pas vide,
* $login : le nom s’utilisateur de la personne connectée (ici on fait une vérification plus tard dans le code) ;

Etant donné que notre installation des différents programmes se fait automatiquement sur le RaspberryPi, nous avons un fichier « installed\_path.txt », qui contient le chemin absolu vers le dossier où sont stockées les photos liées aux captures. Il faut donc lire ce fichier depuis le site pour renseigner, ensuite ce chemin dans la base. C’est ce qui est fait en lignes 23 à 25 :

* D’abord on ouvre le fichier et on stocke son contenu dans « $myfile »,
* Ensuite, on lit cette variable pour en extraire le chemin et le stocker à son tour dans une autre variable, ici, « $python\_path »,
* Enfin on ferme ce fichier grâce à la fonction « close() » ;

Pour le stockage des photos dans la base de données, nous avons décidé de prendre la photo, stockés son chemin (via le programme Python) et après de mettre à jour la dernière ligne de la base pour y rentrer les informations liées à cette prise.

Pour exécuter le programme Python depuis le script PHP, nous avons utilisé la fonction « *exec()* » qui permet de simuler une ligne de commande dans le terminal. Nous avons donc appelé un script en Bash qui lui lancera le programme, et ce grâce a la ligne 28. Aussi, étant donné que le script Python prend un peu de temps à s’exécuter, nous avons préféré faire attendre le script PHP avant de mettre à jour la base de données, d’où la ligne 29.

Ensuite, nous faisons une requête SQL simple pour récupérer le numéro correspondant aux photos misent dans la base : « *SELECT id\_photo FROM photo* », et il nous suffit de prendre le dernier indice du tableau renvoyé pour avoir le numéro de la dernière photo (l.39).

Puis, nous mettons à jour la base de données, par le biais des requêtes SQL « *UPDATE … SET* », lignes 50 et 57.

Mais avant cela on vérifie s’il y a un utilisateur connecté. Pour cela, on vérifie s’il y a une variable de session correspondante à l’utilisateur connecté (l.40). Si c’est le cas :

* On fait une requête pour récupérer l’*id* de cet utilisateur,
* Et ensuit on l’associe a la photo en rajoutant sont *id* dans le champ *#user\_photo* de la ligne correspondante a la photo (l.50).

Si aucun utilisateur n’est connecté, on laisse le champs vide.

Et enfin on renvoie l’utilisateur à la page d’accueil, grâce à la fonction « header() », à la ligne 63.

# Retour d’expérience personnelles

### Retour de Jean-Baptiste

Durant le second semestre, je me suis occupé de l’organisation en mettant en place le GitHub. J’ai beaucoup apprécié d’avoir pu utiliser ce logiciel de versionning puisqu’il très utilisé dans le monde professionnel. J’ai aussi aimé la liberté que l’on nous a donné pour le choix des langages que nous pouvions utiliser pour les différentes fonctions de notre solution. Finalement, j’ai beaucoup apprécié ce projet qui nous a permis de se rendre compte de tache que représente la réalisation d’un projet de A à Z dans une entreprise.

### Retour de Mael

Cette partie a été beaucoup plus simple pour moi, par le fait que c’était plus applicatif. Je préfère passer des nuits à coder que de passer des journées à faire de la gestion de projet. De plus, j’ai pu apprendre a bien maîtrisé un outil de gestion de projet : GitHub. Grâce à celui-ci nous avons une trace de chaque étape du projet.

Ensuite, je n’ai pas eu le temps de m’occuper de la partie qui m’a été assignée au début du projet. Cela est due à la désertion d’un membre du groupe. En effet, celui-ci n’a pas effectué le travail qui lui a été assigné et j’avais besoin de sa partie pour faire la mienne. Nous, Nicolas, Jean-Baptiste et moi-même, avons donc décidé de ne pas prendre en compte l’apport de cette personne au groupe, et de ce fait nous avons privilégié le bon fonctionnement de la partie Site Web et passer sur quelque point de l’aspect sécurité de cette dernière. J’essaie de la finir le plus proprement possible pour la fin.

En conclusion, j’ai beaucoup appris pendant ce second semestre, malgré les imprévus, j’espère que notre produit vous plaira.

### Retour de Nicolas

Cette partie a été pour moi très enrichissante, j’ai appris a utilisé GitHub et j’ai enfin pu voir en application des codes écrit en python par mon groupe et moi-même, vue qu’il y avait quelqu’un de plus compétant en PHP et en Python mon rôle était plutôt de les soulagés en les aidants. Sinon mon rôle ‘majeur’ dans le groupe était plutôt de tous ce qui était de garder une trace écrite, d’organiser les réunions pour voir l’avancement en bref de tous ce dont on aurait besoin pour écrit ce rapport.

### Retour global du groupe

Cette partie c’est montré beaucoup plus compliqué et demandant en termes de temps pour tout le monde. Bien qu’on eût préparé un Gantt, il nous a été compliqué de respecter les dates prédéfinies à la suite des problèmes que nous avons rencontrés en cours de réalisation du projet. Néanmoins nous avons beaucoup apprécié la réalisation du projet car nous avons pu mettre la plupart de nos idées en œuvre et réaliser une solution concrète.