

# Sommaire

Présentation du système .....	3
Cahier des charges.....	3
Description du système .....	4
Objectifs du projet.....	5
Objectif principal.....	5
Objectifs techniques .....	5
Choix techniques .....	6
Solutions possibles .....	6
Solutions retenues.....	7
Matériel utilisé.....	8
Nichoïr en bois.....	8
Mini-ordinateur Raspberry Pi 4 .....	8
Module Grove Base Hat .....	8
Capteur de température et d'humidité DHT22 .....	9
Cellule de charge micro CZL611CD.....	9
Capteur de poids HX711 .....	9
Caméra infrarouge AMG8833.....	10
Caméra standard Raspicam v2.1 .....	10
Carte Witty Pi 4 .....	10
Batterie Power Sonic PS1270.....	11
Cellule photovoltaïque Velleman SOL10P .....	11
Contrôleur de charge Steca Solsum 6.6c .....	11
Outils utilisés.....	12
Qt Creator .....	12
Visual Studio Code .....	12
Serveur UwAmp .....	12
GitHub .....	12
Coût du projet.....	13
Répartition des tâches.....	14
Étudiant 1 – Alan Fournier .....	14
Étudiant 2 – Alexis Boutte.....	14
Étudiant 3 – Maksim Konkin.....	14
Diagrammes de Gantt prévisionnels .....	15
Janvier à Février .....	15
Mars à Avril 1 <sup>e</sup> partie .....	16
Mars à Avril 2 <sup>e</sup> partie .....	17
Mai.....	18
UML.....	19
Diagramme des cas d'utilisation.....	19
Diagramme de séquence .....	20
Diagramme de déploiement.....	21

## **Présentation du système**

Un représentant de la LPO Auvergne-Rhône-Alpes est venu nous voir pour exprimer le besoin d'un système permettant le suivi des oiseaux des villes. Notre client est une association comptant plus de 11 440 adhérents et 1500 bénévoles actifs qui effectuent annuellement 300 000 heures de bénévolat au sein de la structure. LPO



**AGIR** pour la  
**BIODIVERSITÉ**  
AUVERGNE-RHÔNE-ALPES

signifie Ligue de Protection des Oiseaux. Bien que son nom suggère qu'il s'agit d'une association s'occupant uniquement de la protection des oiseaux, la LPO agit pour la biodiversité de manière générale. Par exemple, en Auvergne-Rhône-Alpes, la LPO peut très bien mener des actions de tout type, comme des chantiers nature, des sorties sur le terrain, des conférences, des ateliers, des tenues de stand,... Le tout dans un seul but précis, sensibiliser un public large et varié à la protection de la nature.

## **Cahier des charges**

Le cahier des charges du système nous a été communiqué par le client lors de sa demande. Le périmètre du système s'étend aux villes, pour permettre aux bénévoles de l'association de contrôler plus simplement et efficacement la nidification des oiseaux dans le milieu urbain. La première contrainte concerne la forme du système. Il doit s'agir d'un nichoir qui relève la température et l'humidité intérieure et extérieure, qui prend des photos infrarouges et standards des oiseaux, et enfin qui pèse les oiseaux. Le coût final du produit doit être le plus faible possible, pour être profitable à l'association et être installé en grand nombre. Ensuite, pour permettre un domaine de supervision plus étendu, nous devons permettre une consultation des données sur deux supports, une application sur PC ou smartphone et un site web. Bien entendu, le système doit être sécurisé pour éviter aux données de l'association d'être recueillies par des tiers. Concernant les données, celles-ci doivent être stockées à distance, mais aussi localement en cas de panne de connexion. Enfin, la dernière contrainte est celle du temps. Le produit final doit être prêt en mai, ce qui signifie qu'à partir de la date d'expression du besoin, en début janvier, nous avons 5 mois pour produire un système fonctionnel et complet, répondant au cahier des charges explicité ci-dessus.

## **Description du système**

Pour répondre aux besoins du client, nous avons pris comme base un nichoir en bois qui dispose d'un faux fond. Du point de vue technique, nous allons utiliser un mini-ordinateur comme cœur du système. Nous connecterons à cet appareil plusieurs capteurs. Pour commencer, afin de relever la température et l'humidité à l'intérieur et à l'extérieur du nichoir, nous utiliserons deux capteurs qui permettent la mesure simultanée de ces deux grandeurs. Ensuite, le faux fond du nichoir va nous permettre de mettre en place un système de balance relevant le poids des oiseaux. Pour finir, la capture des images à l'intérieur du nichoir sera faite pour que nous puissions distinguer la présence des oiseaux à l'aide d'une caméra infrarouge et d'une caméra standard. Au final, les données nécessaires au suivi de la nidification des oiseaux seront collectées par 5 capteurs différents. Le mini-ordinateur disposera d'un pare-feu pour interdire les connexions extérieures non souhaitées et ainsi sécuriser le système.

Le système enverra ensuite les données à une base de données hébergée sur un serveur distant, pour leur enregistrement. Le système sera sécurisé par un mot de passe fort, et l'accès sera possible uniquement sur le même réseau local. Les valeurs récupérées par les capteurs seront aussi enregistrées directement sur le mini-ordinateur en cas de panne de connexion. Toutes les acquisitions de données seront datées par la carte à leur envoi dans la base de données dans le but de produire des affichages des valeurs en fonction du temps sur les outils de consultation. Cela servira à suivre la progression des relevés de façon claire.

# **Objectifs du projet**

## **Objectif principal**

L'objectif du projet est de concevoir un service d'observation des oiseaux, afin de contrôler leur nidification. La supervision sera faite avec diffusion des informations en temps réel.

Il s'agira donc de disposer d'un nichoir qui comprend des outils de surveillance. Nous travaillerons sur l'acquisition et la mise en forme des observations pour les rendre facilement accessibles par les utilisateurs.

## **Objectifs techniques**

Le premier objectif technique est d'écrire les programmes principaux sur la carte Raspberry qui permettront de récupérer et d'enregistrer les données issues des capteurs dans la base de données distante et locale avec les bons formats.

En amont, il est nécessaire de mettre en place un serveur de base de données distant qui servira à sauvegarder les informations issues des capteurs : température, humidité, poids, et les images des caméras de surveillance.

Nous devons concevoir l'application sur PC et le site web qui serviront tous les deux à récupérer les informations de la base de données distante et de les afficher de façon claire pour les utilisateurs.

Afin de pouvoir utiliser le site web sur le réseau local, nous devons mettre en place un serveur web qui hébergera le site, qui servira à visualiser les données collectées par le nichoir au même titre que l'application sur PC.

Pour que les utilisateurs du système n'aient pas la contrainte d'intervenir de manière régulière dans le but de le recharger, l'ensemble du système faisant les acquisitions doit être autonome en énergie.

Enfin, les équipements du projet doivent être sécurisés en tout point. Cela signifie que les deux outils de consultation, la base de données distante ainsi que la partie physique se trouvant au cœur du nichoir sont concernés par le besoin de sécurité.

## Choix techniques

### Solutions possibles

Le tableau ci-dessous permet d'avoir une vision globale des besoins exprimés par le client et des solutions techniques qui peuvent être réalisées pour les satisfaire.

N° besoin	Besoin	Solutions possibles
1	Contrôler les capteurs	Raspberry Pi 4 Raspberry Pi Zero
2	Récupérer la température et l'humidité relative	DHT22 DHT11
3	Récupérer le poids	HX711 + cellule de charge 780g
4	Capturer des images infrarouges	AMG8833 RPI IR Camera
5	Capturer des images standards	Caméra ToF pour Raspberry Pi 4 Caméra HQ officielle – Monture M12 Raspicam v2.1
6	Gérer l'alimentation du système	Carte Witty Pi 4 Carte Witty Pi 4 Mini
7	Stocker l'énergie nécessaire au fonctionnement du système	Batterie Power Sonic PS1270
8	Contrôler la charge du système	Steca Solsum 6.6c Steca Solsum 8.8c Steca Solsum 10.10c
9	Enregistrer les données à distance	Serveur UwAmp MySQL Serveur MariaDB
10	Enregistrer les données localement	SQLite PostgreSQL
11	Héberger le site web	Serveur web sur Raspberry Serveur web distant

## Solutions retenues

Le tableau ci-dessous présente les solutions qui ont été retenues pour répondre aux besoins et les raisons qui nous ont poussées à faire ces choix.

N° besoin	Solution retenue	Raison
1	Raspberry Pi 4	Ce mini-ordinateur dispose d'une plus grande puissance de calcul que la carte Raspberry Pi Zero.
2	DHT22	Le capteur de température du DHT22 peut relever des valeurs négatives, ce qui est utile durant l'hiver.
3	HX711 + cellule de charge 780g	Il s'agit du seul matériel dont nous disposons déjà. La cellule de poids aurait pu permettre de relever un poids plus élevé, mais 780g suffisent.
4	AMG8833	Il s'agit ici aussi du seul matériel dont nous disposons. Bien que sa résolution soit faible, son mode de fonctionnement est similaire aux capteurs avec lesquels nous avons l'habitude de travailler.
5	Raspicam v2.1	Cette caméra standard est la seule dont nous disposons. Cependant, son mode de fonctionnement est simple, notre choix se serait tout de même porté sur cette solution.
6	Carte Witty Pi 4	La carte Witty Pi 4 Mini est trop petite pour être maintenue entre le Raspberry Pi 4 et le shield, nous avons donc choisi la carte Witty Pi 4, dont le montage convient mieux.
7	Batterie Power Sonic PS1270	Il s'agit de la seule batterie dont nous disposons. Nous aurions pu choisir une batterie 5V, mais l'utilisation d'un contrôleur de charge nous permet d'utiliser une batterie 12V.
8	Steca Solsum 6.6c	Ce contrôleur de charge est le seul que nous avons à disposition. Nous n'avons pas cherché un autre modèle, car celui-ci remplit bien son rôle.
9	Serveur UwAmp MySQL	Nous avons l'habitude d'utiliser le serveur UwAmp. Son interface phpMyAdmin est pratique et claire, contrairement au serveur MariaDB qui fonctionne en lignes de commandes.
10	SQLite	La solution PostgreSQL aurait pu être envisagée au même titre que la base de données SQLite. Ces deux possibilités remplissent leur rôle de façon similaire.
11	Serveur web distant	Le serveur UwAmp distant permet aussi d'héberger facilement un site web. Héberger le site sur la carte Raspberry n'est pas possible puisqu'elle s'éteint lorsqu'elle n'est pas utilisée.

## **Matériel utilisé**

### **Nichoir en bois**

Notre système physique prendra la forme d'un nichoir en bois dont la forme est similaire à l'image qui vous est présentée. Ce nichoir contiendra l'ensemble des capteurs ainsi que le mini-ordinateur qui s'occupera de l'acquisition et de l'enregistrement des données. À proximité du nichoir seront aussi placés les équipements nécessaires à l'autonomie du système tels que la batterie, la cellule photovoltaïque et le contrôleur de charge.



### **Mini-ordinateur Raspberry Pi 4**

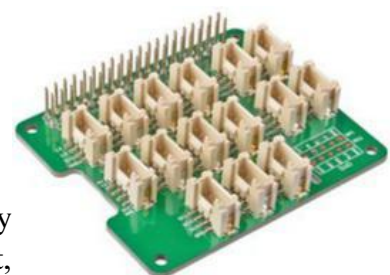
Le cœur du système physique est la carte mini-ordinateur Raspberry Pi 4. La Raspberry Pi 4 se différencie des microcontrôleurs tels que la carte Arduino Uno. En effet, elle est composée d'un microcontrôleur : processeur, mémoire vive et morte, interfaces d'entrées-sorties, ainsi que d'un système d'exploitation, lui valant l'appellation de « mini-ordinateur ».



Au sein du projet, la Raspberry Pi 4 va nous servir à récupérer les données des différents capteurs pour les traiter puis les enregistrer dans la base de données distante. Par ailleurs, elle contiendra une base de données locale pour pouvoir tout de même enregistrer les données dans le cas où la connexion à la base de données distante serait interrompue.

### **Module Grove Base Hat**

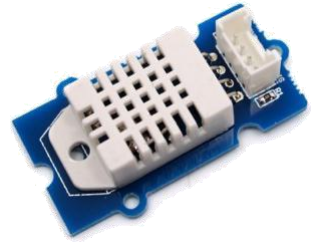
Les connexions entre tous les capteurs et la carte Raspberry Pi 4 sont simplifiées à l'aide du module Grove Base Hat, communément appelé « shield ». Ce module se branche directement sur le mini-ordinateur et met à disposition des ports physiques utilisant différents protocoles de communication en fonction des besoins de l'utilisateur.





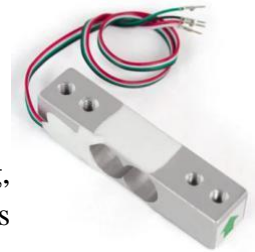
## **Capteur de température et d'humidité DHT22**

Le capteur DHT22 permet de faire l'acquisition de la température et de l'humidité relative de façon simultanée. Sa sensibilité est intéressante vis à vis des besoins du projet puisqu'elle est faible. Les mesures sont effectuées à 0,1°C près pour la température et 0,1 % près pour l'humidité relative. De même, sa plage de relevés est intéressante. Avec ce capteur, il est possible de mesurer la température de -40°C à 80°C, ce qui est idéal pour les relevés en hiver. L'humidité relative est quant à elle relevée de 0 à 100 %. Ce capteur communique avec la carte Raspberry Pi 4 à l'aide d'un port GPIO (General Purpose Input/Output) qui est un port série d'entrées-sorties.



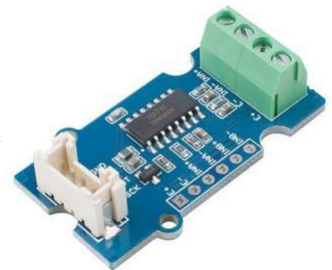
## **Cellule de charge micro CZL611CD**

La cellule de charge utilisée peut mesurer une masse maximale de 780g, ce qui est suffisant en considérant que les oiseaux ne pèsent que quelques grammes. Elle est composée de 4 jauges de contraintes, qui sont des fils dont la résistance varie en fonction de la déformation qui leur est appliquée. Montées électriquement selon le schéma du pont de Wheatstone, il est ainsi possible de récupérer une tension de sortie en fonction de la charge appliquée.



## **Capteur de poids HX711**

Le capteur HX711 récupère la tension en sortie de la cellule de charge, qui sert à la mesure du poids. La tension de sortie est ensuite traitée par le capteur pour être utilisée par la carte Raspberry Pi 4. Le HX711 communique avec le mini-ordinateur à l'aide d'un port série d'entrées-sorties GPIO.



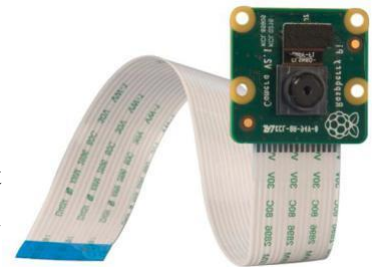
## **Caméra infrarouge AMG8833**



Cette caméra infrarouge permet de capturer des images à l'aide d'un capteur de résolution 8x8 pixels. Cette résolution n'étant pas suffisante pour interpréter les images capturées, il sera nécessaire de mettre en place une interpolation, qui consiste à générer des valeurs intermédiaires entre les valeurs réellement mesurées pour obtenir une meilleure résolution. La caméra infrarouge communique avec la carte Raspberry Pi 4 par l'intermédiaire du bus série I2C.

## **Caméra standard Raspicam v2.1**

Conçue par les mêmes constructeurs que la carte Raspberry Pi 4, cette caméra standard permet de capturer des images en utilisant des lignes de commandes, contrairement au reste des capteurs qui doivent être programmés. Il est possible de régler plusieurs paramètres lors de la prise de l'image tels que la durée d'exposition et le grain par exemple. Cette caméra est reliée au port MIPI CSI situé sur la partie droite du mini-ordinateur. La caméra utilise une nappe et communique à l'aide de l'interface CSI.



## **Carte Witty Pi 4**

La carte Witty Pi 4 gère l'alimentation de la carte Raspberry Pi 4. Elle permet notamment de définir les arrêts et démarrages programmés du système afin de consommer le moins d'énergie possible. Cette fonction est réalisée à l'aide de l'horloge RTC (Real Time Clock) intégrée à la carte qui conserve la date et l'heure en mémoire lorsque le système est éteint. La carte Witty Pi 4 se connecte directement sur la Raspberry Pi 4 de la même manière que le shield. Une fois connectée, l'alimentation doit être branchée sur la carte et non sur le mini-ordinateur.



## **Batterie Power Sonic PS1270**

L'autonomie du système sera assurée par plusieurs éléments, dont la batterie Power Sonic PS1270. Elle délivre une tension de 12V et dispose d'une capacité de 7Ah, ce qui est amplement suffisant compte tenu de l'énergie consommée par le système en moyenne.



## **Cellule photovoltaïque Velleman SOL10P**

Cette cellule photovoltaïque permettra de délivrer l'énergie nécessaire pour recharger la batterie en évitant les pertes. Son rendement est de 11 %, ce qui reste faible. Cependant, en hiver, dans le cas le plus défavorable, cela suffit à recharger suffisamment la batterie pour que le système ne s'éteigne pas.



## **Contrôleur de charge Steca Solsum 6.6c**

Afin de relier tous les éléments permettant de mettre en autonomie le système, nous aurons besoin du contrôleur de charge Steca Solsum 6.6c. Dessus, nous connecterons la batterie, la cellule photovoltaïque et la carte Witty Pi 4. Notamment, le contrôleur de charge permettra de délivrer les 5V nécessaires en sortie pour le système, avec une batterie de tension 12V.



## **Outils utilisés**

### **Qt Creator**



Afin de développer l'application de consultation sur PC, l'étudiant 2 va se servir de l'environnement de développement Qt Creator. Cet outil est multi-plateforme et dispose d'une multitude de bibliothèques pour mener à bien son projet. Le langage de programmation qui est utilisé est le C++.

### **Visual Studio Code**



Développé par Microsoft, Visual Studio Code est un éditeur de code qui prend en charge un grand nombre de langages de programmation. Notamment, cet outil peut suggérer de compléter notre code, met en évidence la syntaxe, et prend en charge le débogage. Dans le cadre de notre projet, Visual Studio Code a été utilisé avec l'extension Live Server pour développer le site web de consultation en observant les modifications en temps réel.

### **Serveur UwAmp**



Le serveur UwAmp est très simple à mettre en place et à utiliser. Il est disponible sous la forme d'une archive, qu'il suffit de décompresser. Les lettres « Amp » signifient respectivement « Apache », « MySQL » et « PHP ». Ce serveur est composé d'un serveur web Apache pour héberger les sites web, d'un serveur de bases de données MySQL et prend en charge le langage PHP. Cet outil permet donc à la fois d'héberger le site web de consultation ainsi que d'héberger la base de données distante, en proposant une interface claire et simple à utiliser.

### **GitHub**



GitHub est une société qui offre un service d'hébergement de référentiel Git, qui est un système de contrôle de version open source spécifique créé en 2005. Le contrôle de version permet aux développeurs de suivre et gérer les modifications apportées au code d'un projet. Nous avons créé et utilisé un dépôt GitHub pour sauvegarder le code du projet et s'organiser en notant les tâches effectuées dans des fichiers texte.

## Coût du projet

Le coût du projet correspond aux dépenses nécessaires à son fonctionnement. Les dépenses se limitent uniquement au matériel, car l'ensemble des outils et des programmes utilisés sont gratuits. Vous trouverez ci-dessous le prix de chaque élément composant du projet.

Composant	Prix
Nichoir en bois	30€
Raspberry Pi 4	60€
Grove Base Hat	10,6€
DHT22 x2	22,8€
HX711	5,9€
Cellule de charge 780g	7,9€
AMG8833	44,5€
Raspicam v2.1	29,9€
Witty Pi 4	37,95€
Power Sonic PS1270	14,9€
Velleman SOL10P	39,96€
Steca Solsum 6.6c	50,66€
Total	377,87€

Le coût total du matériel composant le projet est égal à 377,87€. Si nous considérons que l'ordinateur qui fait office de serveur est aussi à acheter, nous pouvons alors ajouter au total environ 200€, ce qui ferait un coût de 577,87€ pour l'ensemble. Les prix des composants ont été récupérés sur différents sites de fournisseurs. Il n'a pas été possible de récupérer tous les prix sur le même site.

## **Répartition des tâches**

Pour suivre et organiser l'avancement du projet, nous avons mis en place un dépôt GitHub. Chaque élève dispose d'un fichier de suivi où il explique les tâches qu'il a effectuées, concluantes ou non, durant une session de projet. Ces informations nous permettront de fabriquer le diagramme de Gantt réel, pour le comparer au diagramme de Gant prévisionnel, présenté un peu plus tard dans le dossier commun. Le dépôt GitHub permet aussi de sauvegarder les fichiers et programmes du projet.

### **Étudiant 1 – Alan Fournier**

Alan doit réaliser les programmes principaux sur la carte Raspberry qui permettent l'acquisition et la sauvegarde des données issues des capteurs. Il doit aussi implémenter le processus de prise d'images. Toutes les données récupérées doivent être enregistrées dans une base de données locale et distante. Il s'occupe aussi de mettre en place le service sur le système embarqué qui permet la mise en veille et le réveil programmés. Les tâches qu'il doit effectuer sont à la racine du projet puisque sans les données réelles des capteurs, les autres étudiants ne peuvent pas entièrement compléter leurs tâches.

### **Étudiant 2 – Alexis Boutte**

Alexis s'occupe de la partie Interface Homme Machine (IHM) externe sous forme d'une application pour le contrôle distant du nichoir. Il doit mettre en place un serveur de base de données ainsi que créer la base de données distante qui sert au stockage des informations. Les données doivent pouvoir être affichées et consultées par les utilisateurs. L'application qu'il développe doit pouvoir interpréter les données dans le bon format pour les afficher de façon claire.

### **Étudiant 3 – Maksim Konkin**

Maksim doit mettre en place un serveur web et développe un site Internet hébergé sur ce serveur permettant de visualiser les données collectées par le nichoir. Tout comme Alexis, l'outil de consultation de Maksim doit pouvoir interpréter les données de manière à ce qu'elles paraissent claires aux utilisateurs et contrôler le nichoir.

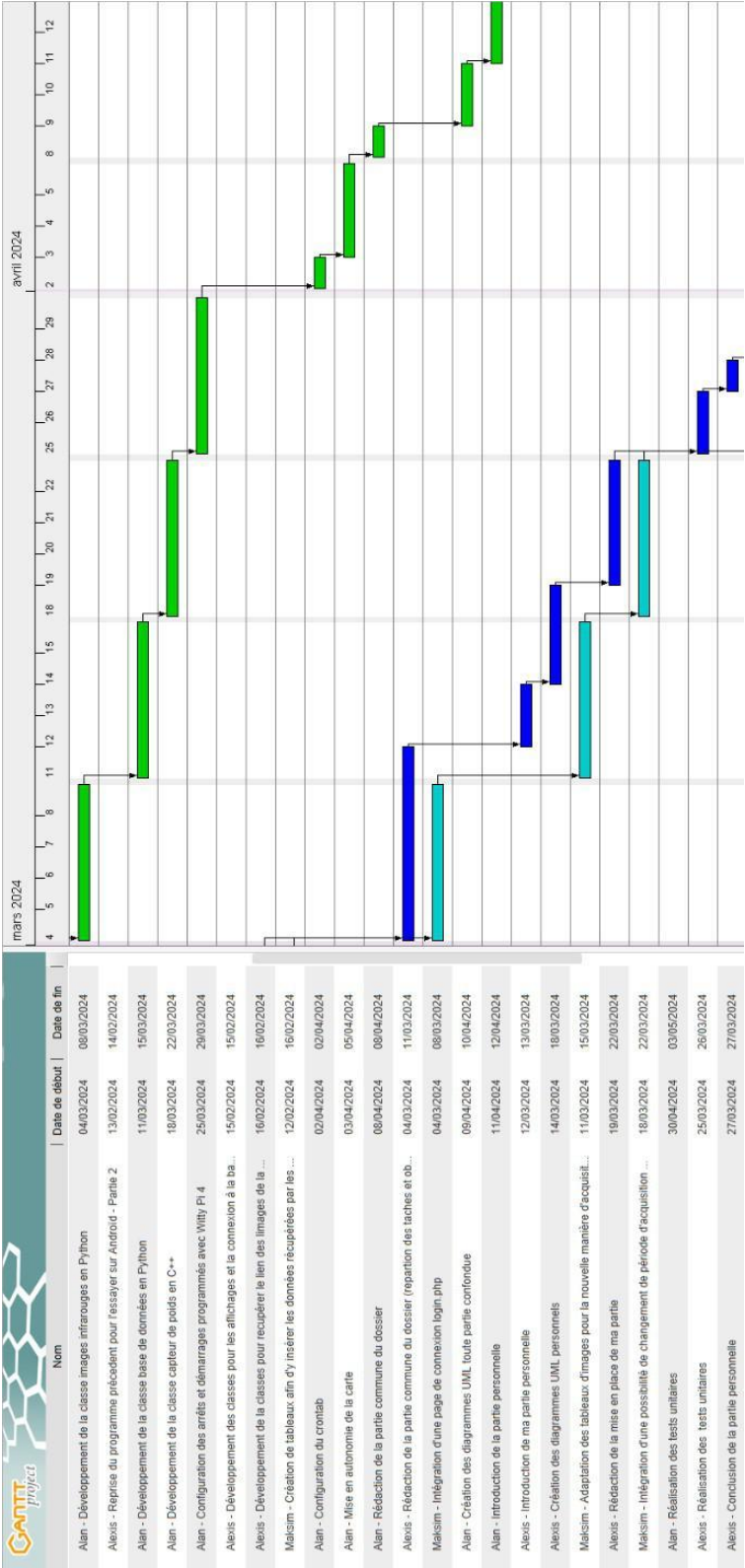
# Diagrammes de Gantt prévisionnels

## Janvier à Février

Les tâches en bleu foncé sont celles d’Alexis, en cyan celles de Maksim et celles en vert sont celles d’Alan.



Mars à Avril 1<sup>e</sup> partie





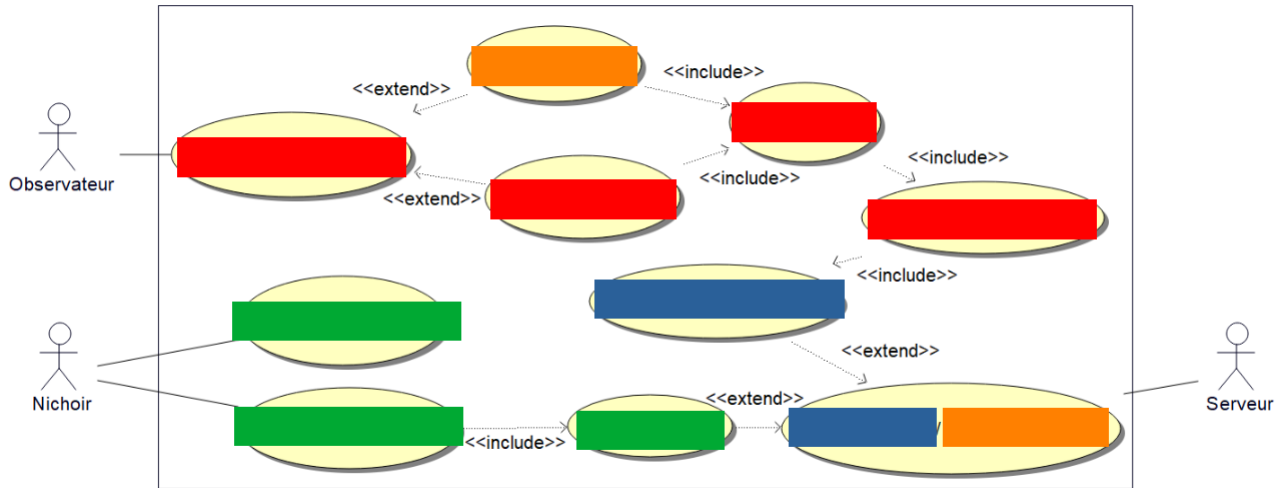
## Mars à Avril 2<sup>e</sup> partie

Alexis - Relecture et correction du dossier personnel	28/03/2024	02/04/2024
Maksim - Création d'une page de style bdd css	25/03/2024	29/03/2024
Alan - Rédaction de la mise en place matérielle et logicielle du système	13/05/2024	14/05/2024
Alan - Rédaction de la conclusion de la partie personnelle	15/05/2024	16/05/2024
Alan - Mise en commun du dossier	17/05/2024	17/05/2024
Alan - Relecture et correction du dossier complet	21/05/2024	22/05/2024
Alan - Création du diaporama de présentation et entraînement	23/05/2024	28/05/2024
Alexis - Création du diaporama et entraînement	03/04/2024	05/04/2024
Maksim - Intégration d'une possibilité de suppression de données au choix à p...	02/04/2024	05/04/2024
Maksim - Travail sur la partie personnelle	08/04/2024	12/04/2024
Maksim - Création de diaporama et entraînement à l'oral	29/04/2024	07/05/2024



# UML

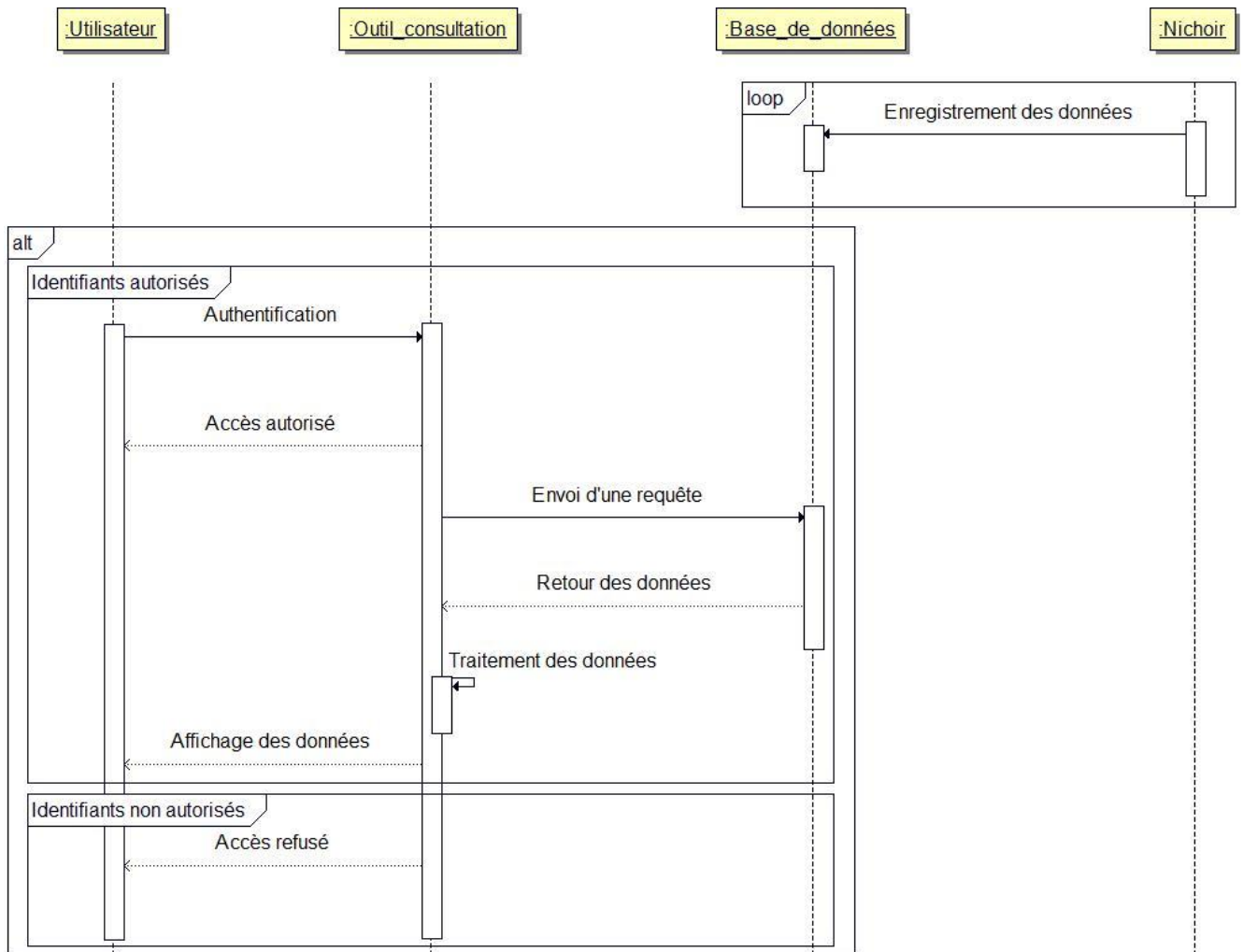
## Diagramme des cas d'utilisation



Les zones vertes représentent les tâches effectuées par Alan, l'étudiant 1. Les zones bleues représentent les tâches du projet effectuées par Alexis, l'étudiant 2. Les tâches effectuées par Maksim, l'étudiant 3, sont représentées par les zones oranges. Enfin, les zones rouges représentent les tâches effectuées en commun par Alexis et Maksim, les étudiants 2 et 3.

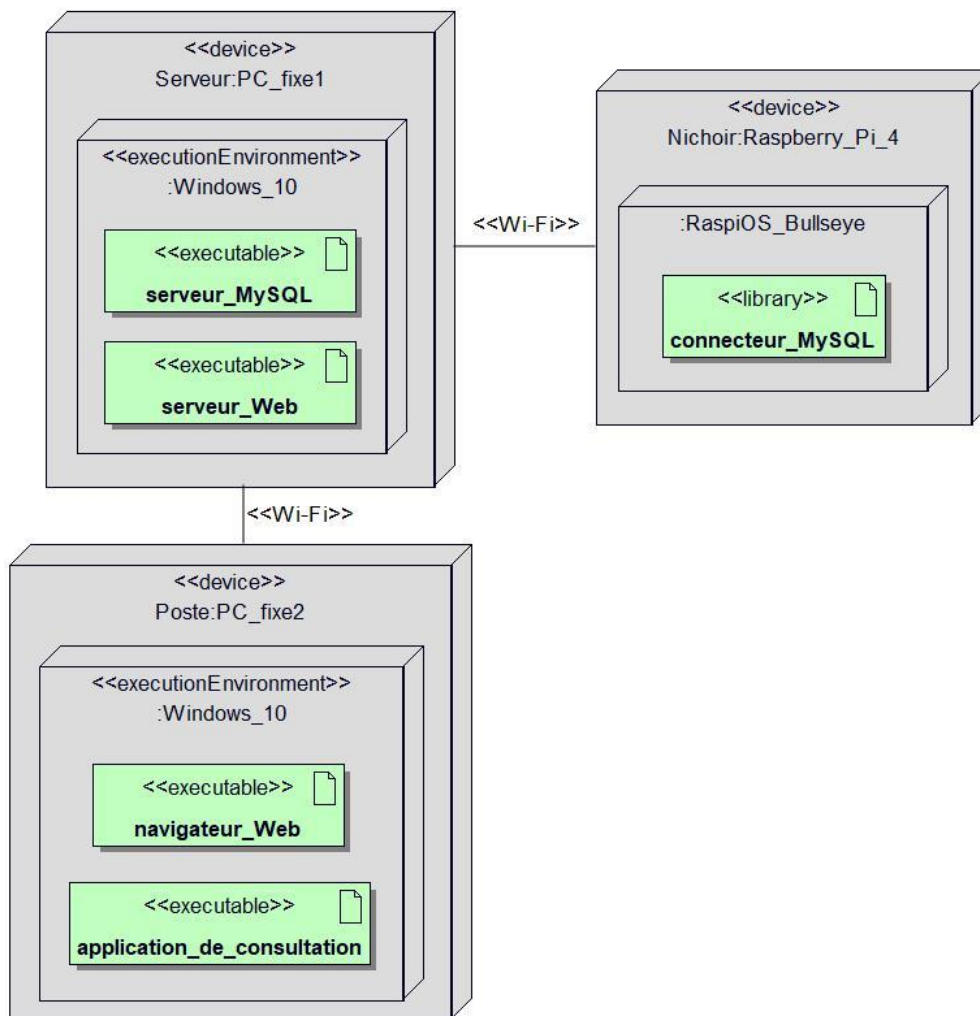
Lorsque l'observateur souhaite consulter les données, elles doivent au préalable avoir été collectées par le nichoir, puis sauvegardées sur le serveur de base de données. Les données sont ensuite affichées sur l'application ou sur le site web en fonction de la méthode souhaitée par l'observateur.

## Diagramme de séquence



Si l'utilisateur dispose d'identifiants autorisés, il peut accéder aux données, l'outil de consultation envoie alors une requête à la base de données. Une fois récupérées, les données sont traitées puis affichées par l'outil de consultation. À l'inverse, si ses identifiants de l'utilisateur ne sont pas autorisés, l'utilisateur ne peut pas accéder aux données. L'outil de consultation lui indique que l'accès lui est refusé.

## Diagramme de déploiement



La carte Raspberry Pi 4 communique en Wi-Fi avec la base de données distante pour enregistrer les données des capteurs qu'elle récupère. Sur un PC fixe, nous pourrions exécuter une application de consultation qui se connectera à la base de données en Wi-Fi pour accéder aux données et les afficher, ou alors utiliser un navigateur web pour consulter les données sur un site Internet.

# **Partie personnelle : Étudiant 3 – Maksim Konkin**

## **Sommaire :**

Partie personnelle : Étudiant 3 – Maksim Konkin .....	1
Objectif: .....	2
Logiciels utilisé : .....	2
Langages de programmation utilisé : .....	3
Schéma de fonctionnement .....	4
Diagramme de cas d'utilisation : .....	5
Diagramme de séquence : .....	6
Analyse du code : .....	7
config.php : .....	7
login.php : .....	7
index.php : .....	9
del.php : .....	11
style.css : .....	12
Conclusion : .....	13
Améliorations possibles: .....	13



## Objectif:

Mon objectif principal dans ce projet était de développer un site Internet qui permet la visualisation de données collectées par le nichoir. Cela inclus les retours des capteurs ainsi que des images prises par les caméras IR. Le site doit être sécurisé afin d'éviter des fuites de données. Il devra aussi permettre aux utilisateurs de s'identifier.

## Logiciels utilisé :



**UwAmp :** UwAmp fournit un environnement de serveur local complet comprenant Apache, MySQL, PHP et PHPMyAdmin, qui sont les composants essentiels pour le développement de la plupart des sites web dynamiques. Cela permet aux développeurs de travailler dans un environnement qui simule leur serveur de production, assurant ainsi que ce qui fonctionne localement fonctionnera également en ligne.

Afin de développer le site web, j'ai dû déployer UwAmp localement pour tester et déboguer mon code en temps réel sans avoir à transférer constamment des fichiers vers un serveur distant. Pour ce faire, il m'a fallu placer les fichiers du site dans le répertoire \www de UwAmp. Ce répertoire est le document root par défaut d'Apache. Par la suite, j'accédais au site via un navigateur en entrant `http://localhost`.

### **Avantages de UwAmp :**

- **Portabilité:** UwAmp peut être exécuté depuis une clé USB, ce qui le rend idéal pour les développeurs en déplacement.
- **Simplicité:** Son installation et sa configuration sont simplifiées par rapport à d'autres environnements similaires.
- **Polyvalence:** Supporte plusieurs versions de PHP et offre une flexibilité dans la configuration d'Apache et MySQL.
- **Gratuité:** UwAmp est gratuit, ce qui en fait une solution accessible pour les développeurs de tous niveaux.

En résumé, UwAmp est un outil puissant et pratique pour le développement local de sites web. Ce logiciel m'a été indispensable tout au long de ce projet.



**Visual Studio Code :** Visual Studio Code (VS Code) est un éditeur de code source léger mais puissant, développé par Microsoft. Il supporte une large gamme de langages de programmation et offre de nombreuses extensions pour améliorer l'expérience de développement.

Pour créer le site web, j'ai utilisé Visual Studio Code. J'ai pu configurer mon environnement de développement avec les extensions nécessaires pour travailler avec HTML, CSS, JavaScript, et PHP. De plus, j'ai utilisé l'extension Live Server pour lancer un serveur de développement local, permettant de recharger automatiquement la page dans le navigateur chaque fois que je sauvegardais un fichier.



## Avantages de Visual Studio Code :

- **Extensibilité:** VS Code dispose d'une vaste bibliothèque d'extensions disponibles dans le Visual Studio Code Marketplace, permettant d'ajouter des fonctionnalités spécifiques à divers langages et frameworks.
- **IntelliSense:** Offre une complétion de code intelligente, la vérification des erreurs en temps réel, et des suggestions automatiques, ce qui améliore la productivité.
- **Intégration Git:** Intégration native avec Git pour le contrôle de version, permettant de gérer les dépôts, de suivre les modifications et de déployer facilement le code.
- **Portabilité:** VS Code est disponible sur Windows, macOS, et Linux, ce qui le rend accessible à tous les développeurs, indépendamment de leur système d'exploitation.
- **Gratuité:** VS Code est gratuit et open-source, ce qui le rend accessible aux développeurs de tous niveaux.
- **Terminal intégré:** Permet d'exécuter des commandes directement depuis l'éditeur sans avoir à basculer entre différentes fenêtres.

Pour résumer, Visual Studio Code est un éditeur de code puissant et flexible qui m'a été indispensable tout au long de ce projet. Grâce à ses nombreuses fonctionnalités et extensions, j'ai pu améliorer mon flux de travail et développer le site web plus rapidement.

## Langages de programmation utilisé :



**HTML :** HTML, acronyme de HyperText Markup Language, est un langage de balisage utilisé pour créer et structurer des pages web. Il utilise des balises pour définir la structure et le contenu d'une page web, ainsi que les liens vers d'autres ressources telles que des images ou des vidéos. En utilisant des balises et des attributs, les développeurs web peuvent organiser le contenu de manière hiérarchique et créer des pages web interactives.



**CSS :** CSS, ou Cascading Style Sheets, est un langage de feuilles de style utilisé pour contrôler la présentation visuelle des pages web écrites en HTML et XML. Plutôt que de définir la structure ou le contenu de la page, comme le fait HTML, CSS spécifie comment les éléments HTML doivent être affichés à l'écran, sur papier ou dans d'autres médias. Avec CSS, les développeurs peuvent définir des styles tels que la couleur, la taille, la police, la disposition et d'autres aspects visuels des éléments HTML. Cela permet une séparation claire entre la structure de la page (définie en HTML) et son apparence (définie en CSS), facilitant ainsi la maintenance du code et permettant des designs flexibles et réutilisables.



**PHP :** PHP, acronyme de "Hypertext Preprocessor", est un langage de programmation populaire et polyvalent, souvent utilisé pour le développement web. PHP est principalement utilisé pour générer du contenu dynamique sur les sites web, comme des pages web générées à partir de bases de données, des formulaires de traitement, des systèmes de gestion de contenu (CMS), des forums en ligne, et bien plus encore.

PHP est interprété côté serveur, ce qui signifie que le code PHP est exécuté sur le serveur web avant d'envoyer le résultat au navigateur du client. Cela permet aux développeurs de créer des sites web interactifs et personnalisés, en utilisant PHP pour interagir avec des bases de données, manipuler des fichiers, envoyer et recevoir des cookies, gérer des sessions utilisateurs, et effectuer de nombreuses autres tâches.

PHP est également connu pour sa facilité d'intégration avec HTML, ce qui permet aux développeurs de mélanger facilement du code PHP avec du code HTML dans le même fichier pour créer des pages web. De plus, PHP est open source, largement supporté et dispose d'une grande communauté de développeurs.

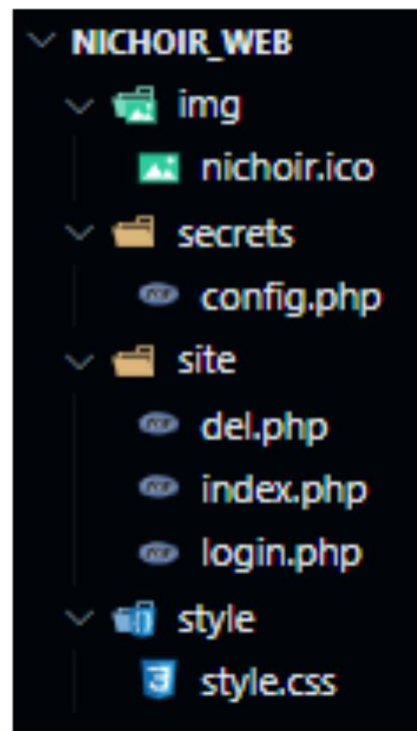
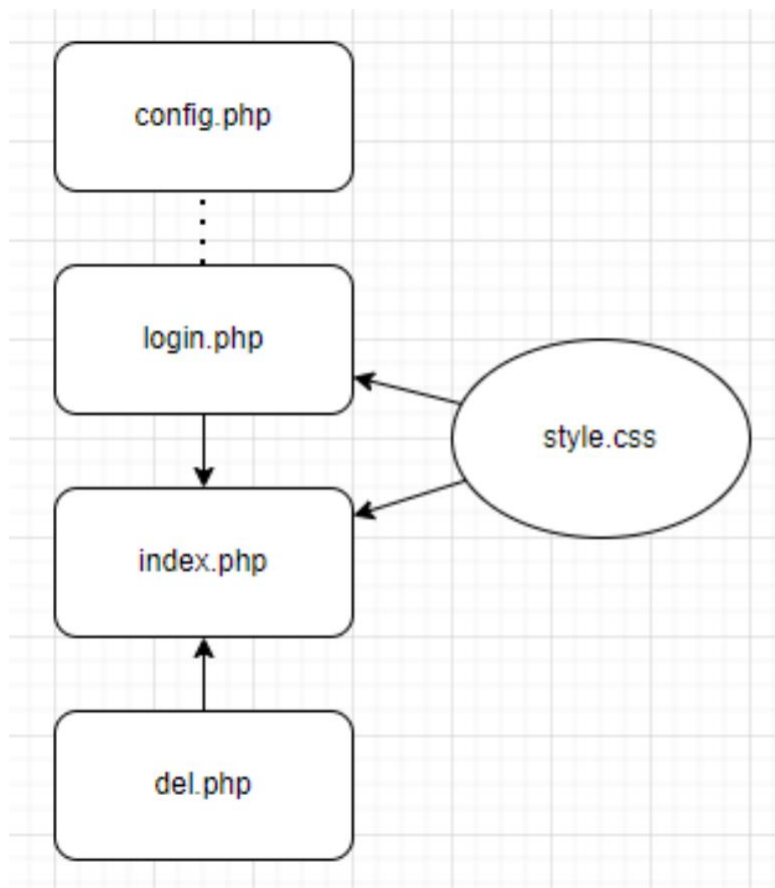


**JS :** JavaScript est un langage de programmation essentiel pour le développement web, permettant d'ajouter de l'interactivité et de la dynamique aux pages web. Il est utilisé pour créer des fonctionnalités telles que des animations, des formulaires interactifs, des jeux et bien plus encore.

En s'exécutant côté client, JavaScript offre des expériences utilisateur fluides et réactives, tandis que son utilisation côté serveur avec des plateformes comme Node.js permet de créer des applications web complètes. Avec sa large adoption et sa vaste communauté de développeurs, JavaScript est un pilier du développement web moderne.

## Schéma de fonctionnement

Voici un schéma simple des fichiers composants le site web :



## Explication du Code PHP

- **config.php** : Ce fichier est crucial pour la configuration de la base de données. Il définit les constantes telles que l'adresse du serveur, le nom d'utilisateur, le mot de passe et le nom de la base de données, facilitant ainsi l'accès à la base de données à partir des autres fichiers PHP.
- **login.php** : Ce fichier sert de page d'authentification. Il contient le code vérifiant les informations d'identification fournies par l'utilisateur. S'il y a correspondance, il démarre une session et redirige vers la page principale. Sinon, il affiche un message d'erreur approprié.
- **index.php** : C'est la page principale du site web. Elle affiche les données des capteurs en temps réel et les images capturées par les caméras. De plus, elle offre la possibilité aux administrateurs de modifier la période d'acquisition des données à travers un formulaire, ainsi que la fonctionnalité permettant de supprimer des données indésirables, comme des images ou des entrées de capteurs de la base de données.
- **del.php** : Ce script gère la suppression des données. En analysant les paramètres dans l'URL, il détermine le type de données à supprimer et exécute la requête SQL appropriée. Il fournit ensuite un retour d'information indiquant si la suppression a été réussie ou non.

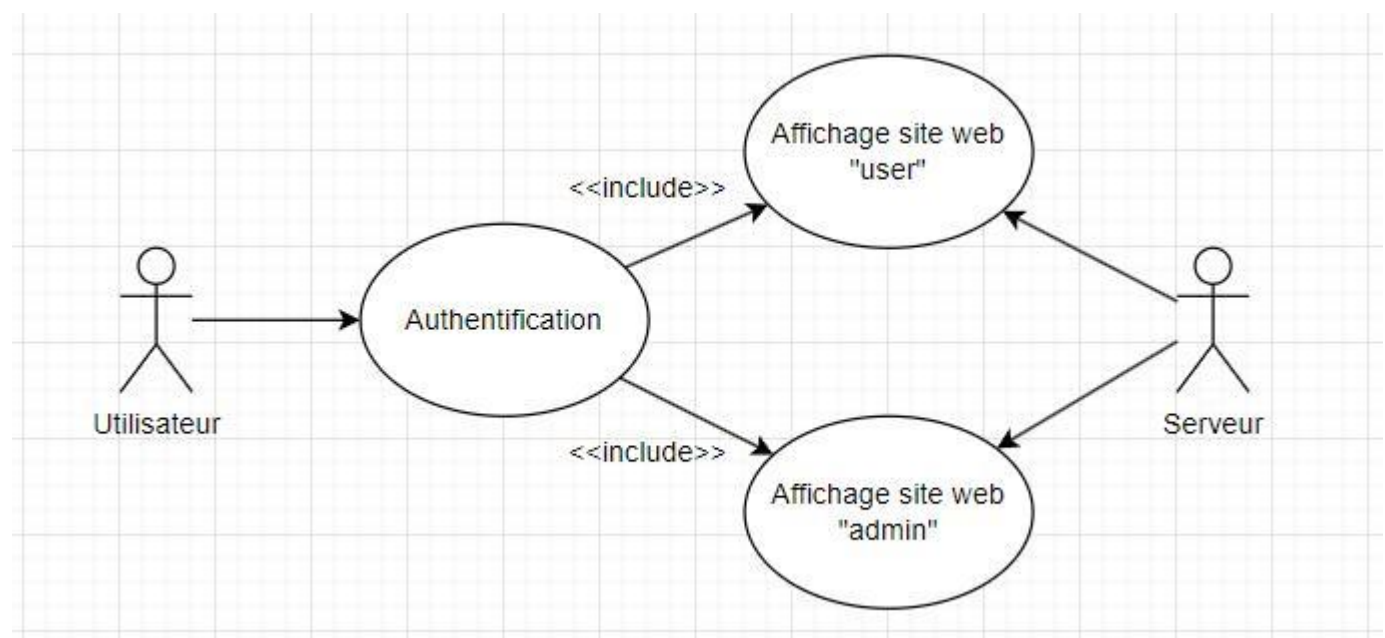
## Explication du Fichier CSS

Le fichier CSS (**style.css**) est responsable de l'apparence du site web. Il définit le style visuel de l'interface utilisateur, et présente esthétiquement les données et les images présentes dans les tableaux.

## Explication du Fichier .ico

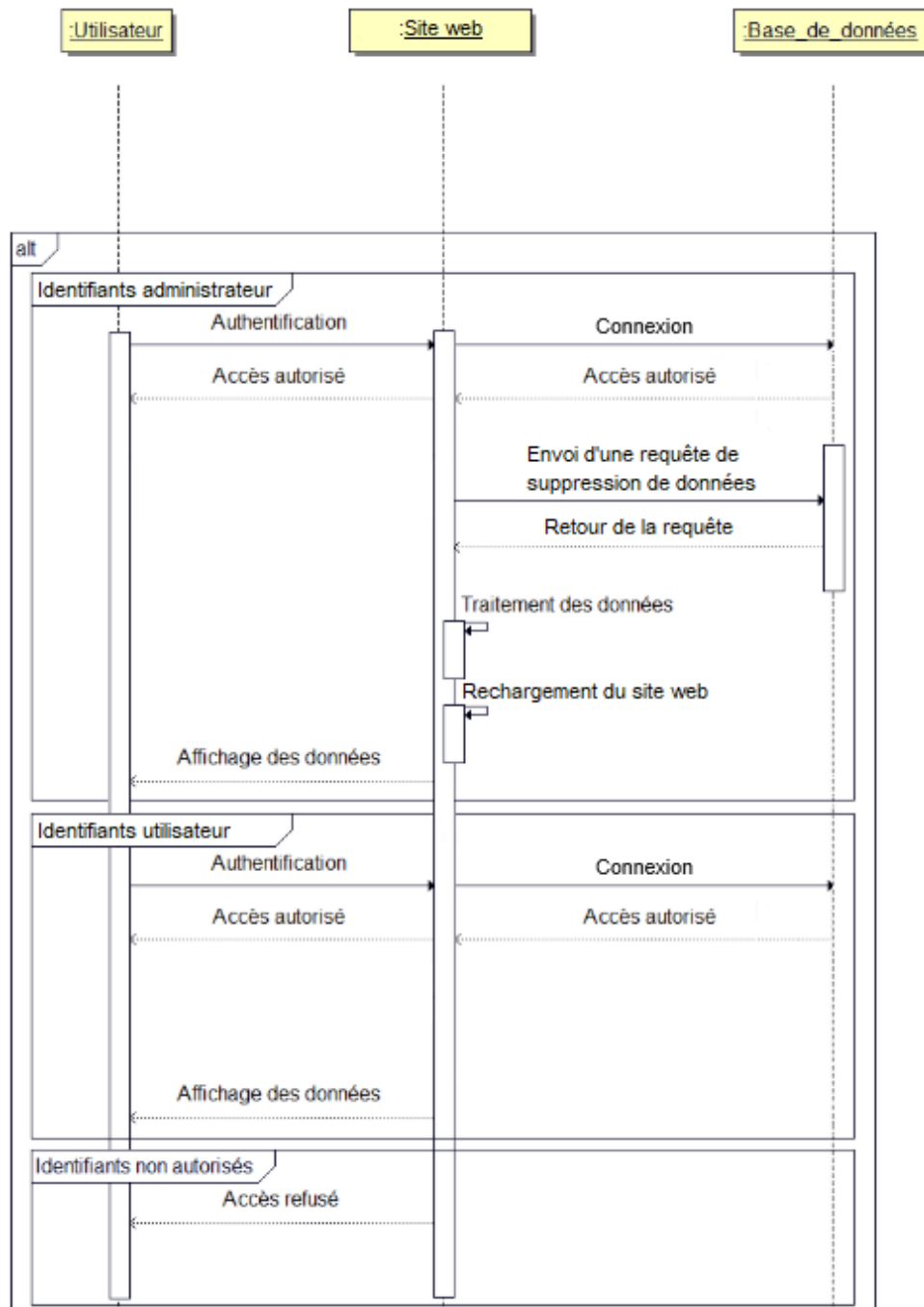
Le fichier nichoir.ico est une icône. Les fichiers **ICO** sont couramment utilisés pour les icônes de site web, dans ce cas c'est une icône représentant un nichoir.

## Diagramme de cas d'utilisation :



Dans ce cas d'utilisation, un utilisateur qui souhaite consulter les données du site devra d'abord s'identifier. Une fois l'utilisateur identifié, le serveur lui enverra la page web en fonction des permissions de l'utilisateur. Si les identifiants admin ont été entrés, la page affichée sera une page administrateur, sur laquelle l'utilisateur pourra modifier les données. Si des identifiants user sont entrés, l'utilisateur pourra seulement consulter les données.

## Diagramme de séquence :



Dans ce diagramme de séquence, l'utilisateur qui souhaite accéder au site web devra passer par une page d'identification. Si l'utilisateur dispose des identifiants, le site web lui sera alors affiché, sinon l'accès lui sera refusé. Si l'utilisateur est administrateur, il pourra supprimer des données de la BDD depuis le site web, ainsi que modifier la période d'acquisition des données. Si l'utilisateur dispose seulement des identifiants utilisateur, il n'aura pas ces permissions et pourra seulement consulter les données.

# Analyse du code :

Dans cette partie on va analyser chaque partie qui compose le site en détail :

## config.php :

```
config.php > ...
1  <?php
2  define('SERVEUR', '172.21.28.52');
3  define('UTILISATEUR', 'root');
4  define('MOT_DE_PASSE', 'tssn2.nichoir!');
5  define('BASE_DE_DONNEES', 'nichoir');
6
```

Le fichier config.php est un élément essentiel pour les applications web en PHP. Il est conçu pour stocker des informations sensibles telles que les paramètres de connexion à la base de données, les clés d'API et d'autres configurations spécifiques à l'application. Voici comment il fonctionne :

- **Définition de constantes:** Les informations sensibles, telles que les noms d'utilisateur, les mots de passe et les noms de base de données, sont stockées en tant que constantes dans le fichier config.php. Ces constantes sont ensuite utilisées dans toute l'application chaque fois que les configurations sont nécessaires.
- **Centralisation des configurations:** Le fichier config.php est utilisé pour centraliser toutes les configurations importantes de l'application en un seul endroit. Cela facilite la gestion et la mise à jour des paramètres.
- **Protection des informations sensibles:** Étant donné que le fichier config.php est placé dans un répertoire sécurisé, les informations sensibles qu'il contient ne sont pas directement exposées aux utilisateurs finaux, ce qui contribue à renforcer la sécurité de l'application.

Le fichier config.php est un composant crucial du site web, il permet de stocker et de gérer de manière sécurisée les configurations nécessaires afin d'accéder à la base de données.

## login.php :

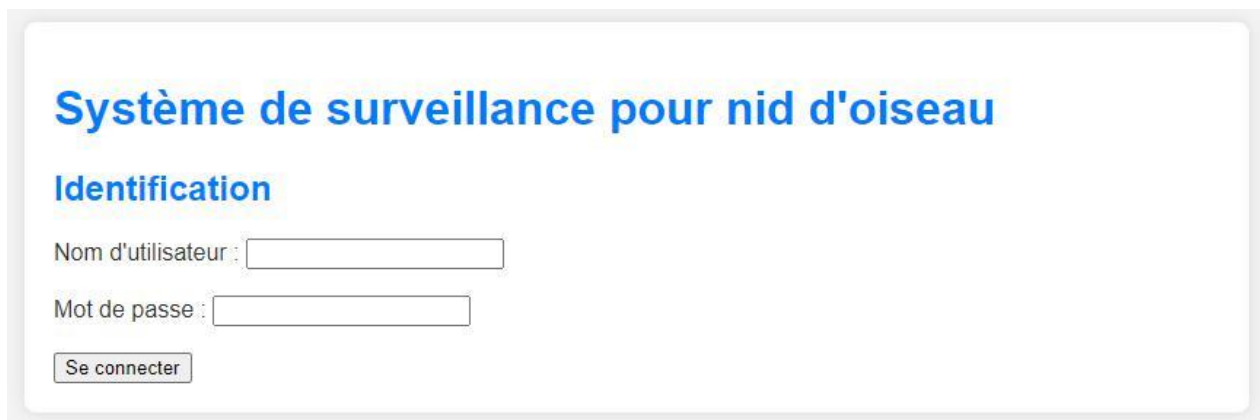
```
1  <?php
2  session_start();
3
4  // Inclure le fichier de configuration
5  require_once('.../secrets/config.php');
6
7  // Connexion à la base de données
8  $connexion = new mysqli(SERVEUR, UTILISATEUR, MOT_DE_PASSE, BASE_DE_DONNEES);
9
10 // Vérifier la connexion
11 if ($connexion->connect_error) {
12     die("Échec de la connexion : " . $connexion->connect_error);
13 }
14
15 // Initialiser la variable d'erreur
16 $error_message = "";
17
18 // Vérification des informations d'identification si le formulaire est soumis
19 if ($_SERVER["REQUEST_METHOD"] == "POST") {
20     // Assurer que les données sont bien définies et non vides
21     if (isset($_POST["username"], $_POST["password"]) && !empty($_POST["username"]) && !empty($_POST["password"])) {
22         // Utiliser des requêtes préparées pour éviter les injections SQL
23         $query = "SELECT * FROM users WHERE u_username = ? AND u_password = ?";
24         $stmt = $connexion->prepare($query);
25
26         // Lier les paramètres
27         $stmt->bind_param("ss", $_POST["username"], $_POST["password"]);
28         $stmt->execute();
29         $result = $stmt->get_result();
30
31         // Vérifier le résultat de la requête
32         if ($result->num_rows == 1) {
33             // Identifiants valides, démarrer une session et rediriger vers la page principale
34             $_SESSION["logged_in"] = true;
35             $_SESSION["username"] = $_POST["username"];
36             header("location: index.php");
37             exit();
38         } else {
39             // Identifiants invalides, définir un message d'erreur
40             $error_message = "Nom d'utilisateur ou mot de passe incorrect.";
41         }
42     } else {
43         // Les champs sont vides, définir un message d'erreur
44         $error_message = "Veuillez saisir un nom d'utilisateur et un mot de passe.";
45     }
46 }
47
48 // Fermer la connexion à la base de données
49 $connexion->close();
50
```

```
52 <!DOCTYPE html>
53 <html lang="fr">
54
55 <head>
56     <meta charset="utf-8" />
57     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
58     <link rel="icon" type="image/x-icon" href=".../img/nichoir.ico">
59     <title>Identification - Nichoir</title>
60     <link rel="stylesheet" type="text/css" href=".../style/style.css">
61 </head>
62
63 <body>
64
65     <div class="container">
66         <h1>Système de surveillance pour nid d'oiseau</h1>
67         <h2>Identification</h2>
68         <form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?">
69             <div>
70                 <label for="username">Nom d'utilisateur :</label>
71                 <input type="text" id="username" name="username" required>
72             </div>
73             <br>
74             <div>
75                 <label for="password">Mot de passe :</label>
76                 <input type="password" id="password" name="password" required>
77             </div>
78             <br>
79             <button type="submit">Se connecter</button>
80         </form>
81         <?php if (!empty($error_message)) { ?>
82             <p style="color: red;"><?php echo $error_message; ?></p>
83         <?php ?>
84     </div>
85 </body>
86
87 </html>
```

Ce code login.php est responsable de l'authentification des utilisateurs . Voici comment il fonctionne :

- **Démarrage de la session PHP** : La première ligne `session_start()` initialise une session PHP, permettant de stocker des variables de session pour gérer l'authentification de l'utilisateur.
- **Inclusion du fichier de configuration** : Le fichier `config.php` est inclus pour accéder aux informations de connexion à la base de données.
- **Connexion à la base de données** : Le code établit une connexion à la base de données en utilisant les constantes définies dans `config.php`.
- **Validation du formulaire** : Lorsque le formulaire de connexion est soumis (`$_SERVER["REQUEST_METHOD"] == "POST"`), le code vérifie si les champs `username` et `password` sont définis et non vides.
- **Requête SQL sécurisée** : Une requête SQL préparée est utilisée pour éviter les attaques par injection SQL. Les identifiants saisis par l'utilisateur sont comparés à ceux stockés dans la base de données.
- **Authentification de l'utilisateur** : Si les identifiants sont valides, une session est démarrée pour l'utilisateur et il est redirigé vers la page principale (`index.php`).
- **Gestion des erreurs** : Si les identifiants sont incorrects ou si les champs sont vides, un message d'erreur approprié est affiché pour informer l'utilisateur.
- **Fermeture de la connexion à la base de données** : Une fois l'authentification terminée, la connexion à la base de données est fermée pour libérer les ressources.

Ce code assure l'authentification sécurisée des utilisateurs en vérifiant leurs identifiants par rapport à ceux stockés dans la base de données, tout en évitant les vulnérabilités potentielles liées aux injections SQL.



The screenshot shows a web form titled "Système de surveillance pour nid d'oiseau" in blue. Below the title is a section header "Identification" in blue. The form contains two input fields: "Nom d'utilisateur :" and "Mot de passe :". Below these fields is a button labeled "Se connecter". The entire form is enclosed in a light gray border.

Ici, on voit la page d'authentification.



## index.php :

```

1 <?php
2 session_start();
3
4 // Inclure le fichier de configuration
5 require_once('../secrets/config.php');
6
7 // Connexion à la base de données
8 $connexion = new mysqli(SERVEUR, UTILISATEUR, MOT_DE_PASSE, BASE_DE_DONNEES);
9
10 // Vérifier la connexion
11 if ($connexion->connect_error) {
12     die("Échec de la connexion : " . $connexion->connect_error);
13 }
14
15 $is_admin = false;
16
17 // Vérifier si l'utilisateur est connecté
18 if (isset($_SESSION["logged_in"]) || $_SESSION["logged_in"] != true) {
19     header("Location: login.php");
20     exit;
21 }
22
23 // Vérifier si l'utilisateur est admin
24 if (isset($_SESSION["username"]) && $_SESSION["username"] === "admin") {
25     $is_admin = true;
26 }
27
28 }
29 <!DOCTYPE html>
30 <html lang="fr">
31
32 <head>
33     <meta charset="utf-8" />
34     <meta name="viewport" content="width=device-width" />
35     <link rel="icon" type="image/x-icon" href="../img/nichoir.ico">
36     <title>Page principale - Nichoir</title>
37     <link rel="stylesheet" type="text/css" href="../style/style.css">
38 </head>
39
40 <body>
41
42     <div class="container">
43         <h1>Bienvenue, <?php echo htmlspecialchars($_SESSION["username"]); ></h1>
44         <?php
45         if ($is_admin) {
46             echo "<form method='post' action='../.htmlspecialchars($_SERVER["PHP_SELF"]) . ">";
47             <label for="u_sched">Changer la période d'acquisition :</label>
48             <select id="u_sched" name="u_sched">
49                 <option value="1">30 minutes</option>
50                 <option value="2">1 heures</option>
51                 <option value="3">2 heures</option>
52             </select>
53             <button type="submit">Changer</button>
54         </form>;

```

```

180 // Récupération des 5 dernières données du tableau d0122_int avec les identifiants primaires
181 // d0122_int, d0122_int - 1
182 $resultat_d0122_int = $connexion->query("SELECT d01_jd, d1_date_heure, d1_temperature, d1_humidite FROM d0122_int ORDER BY d1_date_heure DESC LIMIT 5");
183
184 if ($resultat_d0122_int->num_rows > 0) {
185     echo "02Données du capteur d0122_int lue(s) :<br>";
186     echo "étiquette horaire:1'5'";
187     echo "<br>";
188     if ($is_admin) {
189         echo "<td>id(tro)</td>";
190         echo "<td>superlamer(tro)</td>";
191     }
192     echo "<td>date/heure(tro)<td>temperature(tro)<td>humidite(tro)</td>";
193     while ($row = $resultat_d0122_int->fetch_assoc()) {
194         $temperature_int = $row["d1_temperature"] . " °C";
195         $humidite_int = $row["d1_humidite"] . " %";
196         echo "<tr>";
197         if ($is_admin) {
198             echo "<td>". $row["d01_jd"] . "</td>";
199             echo "<td>". $row["d1_date_heure"] . "</td>";
200             echo "<td>". $row["d1_temper"] . "</td>";
201             echo "<td>". $row["d1_humid"] . "</td>";
202             echo "<td>". $row["d1_date_heure"] . "</td>";
203             echo "<td>". $temperature_int . "</td>";
204             echo "<td>". $humidite_int . "</td>";
205         }
206         echo "<tr>";
207     }
208 } else {
209     echo "Aucune donnée trouvée pour le capteur d0122_int.";
210 }
211
212
213 // Récupération des 5 dernières données du tableau h0711 avec les identifiants primaires
214 $resultat_h0711 = $connexion->query("SELECT h01_jd, h1_date_heure, h1_poids FROM h0711 ORDER BY h1_date_heure DESC LIMIT 5");
215
216 if ($resultat_h0711->num_rows > 0) {
217     echo "02Données du capteur h0711 :<br>";
218     echo "étiquette horaire:1'5'";
219     echo "<br>";
220     if ($is_admin) {
221         echo "<td>id(tro)</td>";
222         echo "<td>superlamer(tro)</td>";
223     }
224     echo "<td>date/heure(tro)<td>Poids(tro)</td>";
225     while ($row = $resultat_h0711->fetch_assoc()) {
226         $poids_h0711 = $row["h1_poids"] . " g";
227         echo "<tr>";
228         if ($is_admin) {
229             echo "<td>". $row["h01_jd"] . "</td>";
230             echo "<td>". $row["h1_date_heure"] . "</td>";
231             echo "<td>". $poids_h0711 . "</td>";
232         }
233         echo "<tr>";
234     }
235 } else {
236     echo "Aucune donnée trouvée pour le capteur h0711.";
237 }

```

[illegible]

```

15 // Récupération des 5 dernières données du tableau images_stand avec les identifiants primaires
16 $Resultat_images_stand = $connexion->query("SELECT id_pk_id, id_date_heure, id_img FROM images_stand ORDER BY id_date_heure DESC LIMIT 5");
17
18 echo "<div>images_stand -> num_row = 49 </div>";
19
20 echo "<div>images_stand -> id2/3 </div>";
21
22 echo "<table border='1'>";
23
24 echo "<tr>";
25
26 if ($id_images) {
27     echo "<td>id2/3/3b/";
28     echo "</td></tr>";
29 }
30
31 echo "<td>id3/superImage/3b/";
32
33 }
34
35 echo "<td>id4/date/3b/";
36
37 }
38
39 while ($row = $Resultat_images_stand->fetch_assoc()) {
40     $image_date = $row['id4/date'];
41     $img_id = $row['id3'];
42     $image_url = "data:image/jpeg;base64, " . $image_data;
43     echo "<tr>";
44     if ($id_images) {
45         echo "<td>";
46         $row["id3/id3"];
47         echo "</td>";
48         echo "<td>id4/date=" . $urlid3->superImageBasename("$images_stand", " - $row[id3/id3", " ]>superImage/3b/";
49         echo "</td>";
50         $row["id4/date"];
51         echo "<td>";
52         $row["id4/date_heure"];
53         echo "</td>";
54         echo "<td>";
55         $row["id3/id3"];
56         echo "</td>";
57     }
58     echo "<td>";
59     $row["id4/date"];
60     echo "</td>";
61 } else {
62     echo "Aucune image standard trouvée.";
63 }
64
65 // Récupération des 5 dernières données du tableau images_infra avec les identifiants primaires
66 $Resultat_images_infra = $connexion->query("SELECT id_pk_id, id_date_heure, id_img FROM images_infra ORDER BY id_date_heure DESC LIMIT 5");
67
68 echo "<div>images_infra -> num_row = 49 </div>";
69
70 echo "<div>images_infra -> id2/3 </div>";
71
72 echo "<table border='1'>";
73
74 echo "<tr>";
75
76 if ($id_images) {
77     echo "<td>id2/3/3b/";
78     echo "</td></tr>";
79 }
80
81 echo "<td>id3/superImage/3b/";
82
83 }
84
85 echo "<td>id4/date/3b/";
86
87 }
88
89 while ($row = $Resultat_images_infra->fetch_assoc()) {
90     $image_date = $row['id4/date'];
91     $img_id = $row['id3'];
92     $image_url = "data:image/jpeg;base64, " . $image_data;
93     echo "<tr>";
94     if ($id_images) {
95         echo "<td>";
96         $row["id3/id3"];
97         echo "</td>";
98         echo "<td>id4/date=" . $urlid3->superImageBasename("$images_infra", " - $row[id3/id3", " ]>superImage/3b/";
99         echo "</td>";
100         $row["id4/date"];
101         echo "<td>";
102         $row["id4/date_heure"];
103         echo "</td>";
104         echo "<td>";
105         $row["id3/id3"];
106         echo "</td>";
107     }
108     echo "<td>";
109     $row["id4/date"];
110     echo "</td>";
111 } else {
112     echo "Aucune image infrarouge trouvée.";
113 }
114
115 // Fin de la connexion à la base de données
116 $connexion->close();
117
118

```

```

215 <script>
216 // Fonction pour envoyer la requête de suppression en utilisant AJAX
217 function supprimerDonnee(type, id) {
218     if (confirm("Voulez-vous vraiment supprimer cette donnée ?")) {
219         fetch('del.php?type=${type}&id=${id}')
220             .then(response => response.text())
221             .then(data => {
222                 if (data === "success") {
223                     // Recharger la page après la suppression réussie
224                     location.reload();
225                 } else {
226                     // Afficher un message d'erreur sans recharger la page
227                     alert("Erreur lors de la suppression de la donnée : " + data);
228                 }
229             })
230             .catch(error => {
231                 console.error('Erreur lors de la suppression:', error);
232                 // Afficher un message d'erreur sans recharger la page
233                 alert("Une erreur s'est produite lors de la suppression de la donnée.");
234             });
235     }
236 }
237 </script>
238 </body>
239
240 </html>

```

Le fichier index.php est la page principale du site web, elle fournit un tableau de bord pour surveiller et gérer les données récupérées depuis la base de données. Voici une explication détaillée de son fonctionnement :

- **Initialisation de la session et connexion à la base de données** : Le script commence par démarrer une session PHP avec `session_start()`. Ensuite, il inclut le fichier de configuration `config.php` qui contient les informations de connexion. Ensuite, il établit une connexion à la base de données en utilisant les informations fournies.
- **Vérification de l'authentification utilisateur** : Le script vérifie si l'utilisateur est authentifié en vérifiant la variable de session `$_SESSION["loggedin"]`. Si l'utilisateur n'est pas authentifié, il est redirigé vers la page de connexion `login.php`.
- **Détermination des privilèges administratifs** : Le script vérifie si l'utilisateur est un administrateur en vérifiant le nom d'utilisateur stocké dans la session. Si l'utilisateur est un administrateur, cela lui permet d'accéder à certaines fonctionnalités supplémentaires.
- **Affichage de la page principale** : Une fois l'authentification vérifiée, le script affiche le contenu de la page principale. Il accueille l'utilisateur avec un message de bienvenue personnalisé et propose une option pour se déconnecter. Si l'utilisateur est un administrateur, il peut également voir un formulaire pour changer la période d'acquisition des données.
- **Affichage des données des capteurs** : Le script récupère les données des capteurs à partir de la base de données et les affiche sous forme de tableaux. Il récupère les 5 dernières données de chaque type de capteur (DHT22 extérieur, DHT22 intérieur, HX711, images standard et images infrarouges) et les affiche dans des tableaux distincts. Si l'utilisateur est un administrateur, chaque entrée de données dans les tableaux est accompagnée d'un lien pour supprimer la donnée correspondante.
- **Suppression des données via AJAX** : Le script utilise JavaScript pour permettre la suppression des données de manière asynchrone via AJAX. Lorsque l'utilisateur clique sur le lien de suppression d'une donnée, une boîte de dialogue de confirmation apparaît. Si l'utilisateur confirme la suppression, une requête est envoyée à `del.php` pour supprimer la donnée de la base de données. Si la suppression réussit, la page est rechargée pour refléter les changements. Sinon, un message d'erreur est affiché.

index.php est la page principale qui affiche les données des capteurs du système d'oiseaux, il permet à l'utilisateur de voir les données, de se déconnecter et, si l'utilisateur est un administrateur, de modifier la période d'acquisition des données et de supprimer des données.

**Bienvenue, admin!**  
Changer la période d'acquisition : 30 minutes   
[Se déconnecter](#)

**Système de surveillance pour nid d'oiseau**  
**Données du capteur DHT22 extérieur :**

ID	Supprimer	Date/Heure	Température	Humidité
9	<a href="#">Supprimer</a>	2024-05-27 12:45:00	25.9 °C	49.8 %
8	<a href="#">Supprimer</a>	2024-05-27 12:30:00	26.3 °C	52.1 %
7	<a href="#">Supprimer</a>	2024-05-27 12:15:00	24.8 °C	48.5 %
6	<a href="#">Supprimer</a>	2024-05-27 12:00:00	25.5 °C	50 %

**Bienvenue, user!**  
[Se déconnecter](#)

**Système de surveillance pour nid d'oiseau**  
**Données du capteur DHT22 extérieur :**

Date/Heure	Température	Humidité
2024-05-27 12:45:00	25.9 °C	49.8 %
2024-05-27 12:30:00	26.3 °C	52.1 %
2024-05-27 12:15:00	24.8 °C	48.5 %
2024-05-27 12:00:00	25.5 °C	50 %



Sur la première capture d'écran, l'utilisateur est authentifié en tant qu'administrateur, lui donnant ainsi les permissions administratives. Sur la deuxième capture d'écran, l'utilisateur est authentifié en tant qu'utilisateur standard, ce qui signifie qu'il ne peut que consulter les données et n'a pas accès aux permissions administratives.

## del.php :

```
1  <?php
2  // Vérifier si le type et l'ID sont définis dans l'URL
3  if (isset($_GET["type"]) && isset($_GET["id"])) {
4      // Inclure le fichier de configuration
5      require_once('../secrets/config.php');
6
7      // Connexion à la base de données
8      $connexion = new mysqli(SERVEUR, UTILISATEUR, MOT_DE_PASSE, BASE_DE_DONNEES);
9
10     // Vérifier la connexion
11     if ($connexion->connect_error) {
12         die("Échec de la connexion : " . $connexion->connect_error);
13     }
14
15     // Échapper les données pour éviter les injections SQL
16     $type = $connexion->real_escape_string($_GET["type"]);
17     $id = intval($_GET["id"]); // Convertir en entier pour des raisons de sécurité
18
19     // Préparer la requête de suppression
20     $delete_query = "";
21
22     // Vérifier le type de données et construire la requête de suppression correspondante
23     if ($type === 'dht22_ext') {
24         $delete_query = "DELETE FROM dht22_ext WHERE de_pk_id = ?";
25     } elseif ($type === 'dht22_int') {
26         $delete_query = "DELETE FROM dht22_int WHERE di_pk_id = ?";
27     } elseif ($type === 'hx711') {
28         $delete_query = "DELETE FROM hx711 WHERE h_pk_id = ?";
29     } elseif ($type === 'images_stand') {
30         $delete_query = "DELETE FROM images_stand WHERE is_pk_id = ?";
31     } elseif ($type === 'images_infra') {
32         $delete_query = "DELETE FROM images_infra WHERE ii_pk_id = ?";
33     } else {
34         echo "Type de données non pris en charge.";
35         exit; // Arrêter le script si le type de données n'est pas pris en charge
36     }
37
38     // Préparation de la requête de suppression et liaison des paramètres
39     $stmt_delete = $connexion->prepare($delete_query);
40     $stmt_delete->bind_param("i", $id);
41
42     // Exécution de la requête de suppression
43     if ($stmt_delete->execute()) {
44         echo "success";
45     } else {
46         echo "Erreur lors de la suppression de la donnée : " . $connexion->error;
47     }
48
49     // Fermer la connexion à la base de données
50     $connexion->close();
51 } else {
52     // Si le type ou l'ID n'est pas défini, renvoyer un message d'erreur
53     echo "Paramètres manquants pour la suppression.";
54 }
```

Le fichier del.php est responsable de la suppression des données spécifiées dans la base de données .

Voici comment il fonctionne :

- **Vérification des paramètres** : Le script commence par vérifier si les paramètres type et id sont définis dans l'URL. Ces paramètres sont utilisés pour spécifier le type de données à supprimer (par exemple, dht22\_ext, dht22\_int, hx711, images\_stand, images\_infra) et l'identifiant unique (ID) de la donnée à supprimer.
- **Inclusion du fichier de configuration** : Comme pour login.php, le fichier config.php est inclus pour accéder aux informations de connexion à la base de données.
- **Connexion à la base de données** : Une connexion à la base de données est établie en utilisant les informations de connexion extraites du fichier config.php.

- **Préparation de la requête de suppression** : En fonction du type de données spécifié dans l'URL, une requête SQL de suppression appropriée est construite. Par exemple, si le type est dht22\_ext, la requête sera DELETE FROM dht22\_ext WHERE de\_pk\_id = ?.
- **Exécution de la requête de suppression** : La requête SQL est préparée en utilisant la méthode prepare() pour éviter les injections SQL. Ensuite, l'identifiant unique de la donnée à supprimer est lié à la requête à l'aide de la méthode bind\_param(). La requête est ensuite exécutée à l'aide de la méthode execute().
- **Gestion de la réponse** : Si la requête de suppression s'exécute avec succès, le script renvoie "success" en réponse. Sinon, il renvoie un message d'erreur indiquant le problème rencontré lors de la suppression.
- **Fermeture de la connexion à la base de données** : Une fois la suppression effectuée, la connexion à la base de données est fermée pour libérer les ressources.

Le fichier del.php assure la suppression sécurisée des données spécifiées dans la base de données en utilisant des requêtes préparées pour éviter les attaques par injection SQL. Il fournit une interface simple et sécurisée pour supprimer des données à partir du site web.

## style.css :

```

1  body {
2      font-family: Arial, sans-serif;
3      margin: 0;
4      padding: 0;
5      background-color: #f3f3f3;
6      color: #333;
7  }
8
9  .container {
10     max-width: 800px;
11     margin: 20px auto;
12     padding: 20px;
13     background-color: #fff;
14     border-radius: 8px;
15     box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
16 }
17
18 h1,
19 h2 {
20     color: #007bff;
21 }
22
23 table {
24     width: 100%;
25     border-collapse: collapse;
26     margin-bottom: 20px;
27 }
28
29 table th,
30 table td {
31     padding: 8px;
32     text-align: left;
33     border-bottom: 1px solid #ddd;
34 }
35
36 table th {
37     background-color: #f7f7f7;
38 }
39
40 img {
41     max-width: 100%;
42     height: auto;
43     display: block;
44     margin: 0 auto;
45 }

```

Ce code CSS définit les styles visuels pour la page. Voici ce qu'il fait :

- **body**: Définit les styles de base pour tout le corps de la page. Il spécifie la famille de polices à utiliser (Arial, sans-serif), supprime les marges et les rembourrages par défaut (margin: 0; padding: 0;), définit la couleur de fond (background-color: #f3f3f3;) et la couleur du texte (color: #333;).

- **.container**: Définit les styles pour les éléments avec la classe `.container`. Cet élément est centré horizontalement sur la page avec une largeur maximale de 800 pixels. Il a un remplissage de 20 pixels à l'intérieur, un fond blanc, un bord arrondi de 8 pixels et une ombre portée légère.
- **h1, h2**: Définit les styles pour les titres de niveau 1 (`<h1>`) et de niveau 2 (`<h2>`). Ils ont tous les deux une couleur bleue (`#007bff`);).
- **table**: Définit les styles pour les tableaux. Il spécifie que les tableaux doivent occuper 100% de la largeur disponible, avoir une bordure qui se fusionne (`border-collapse: collapse`;) et avoir une marge en bas de 20 pixels.
- **table th, table td**: Définit les styles pour les cellules d'en-tête et les cellules de données dans les tableaux. Ils auront un remplissage de 8 pixels, un alignement de texte à gauche et une bordure en bas de 1 pixel solide avec une couleur gris clair (`#ddd`);).
- **table th**: Définit les styles spécifiques pour les cellules d'en-tête dans les tableaux. Ils auront un fond gris clair (`#f7f7f7`);).
- **img**: Définit les styles pour les images. Les images seront redimensionnées pour s'adapter à la largeur maximale de leur conteneur (`max-width: 100%`;) tout en conservant leur proportion. Elles seront affichées en bloc avec une marge de 0 pixels sur les côtés et automatiquement centrées horizontalement sur la page.

## Conclusion :

En conclusion, le site est hébergé sur un serveur web et est sécurisé par une page de connexion destinée aux administrateurs et aux utilisateurs, protégée contre les attaques par injection SQL. Le site permet également la visualisation des données collectées par les capteurs en temps réel grâce à une connexion à une base de données. Les administrateurs ont la possibilité de modifier la période d'acquisition des données par les capteurs et de supprimer des données spécifiques.

## Améliorations possibles:

Voici quelques idées d'amélioration pour le site :

- **Alertes et Notifications** : Mettre en place un système d'alertes en temps réel (email, SMS, notifications push) pour informer les utilisateurs de toute anomalie ou événement important détecté par les capteurs.
- **Analyse des Données** : Intégrer des outils d'analyse avancée des données pour permettre aux utilisateurs d'extraire des informations pertinentes.
- **Support Multilingue** : Rendre le site accessible dans plusieurs langues pour rendre le site plus accessible.