

Rapport de Projet Opti'Plant

SAE 6.ESE.01

Omari Alaoui Bilal, Mallem Chakib,
Luppo Mathias, Brunner Baptiste,
Gonon Bastien, Carrière Lilian

BUT 3 GEII ESE B
Année 2024/2025

Sommaire

Introduction	4
I. Partie BDD et requêtes SQL	5
1. Introduction	5
2. Réalisation du MCD/MLD	6
2.1. Modèle Conceptuel de Données (MCD)	6
2.2. Modèle Logique de Données (MLD)	7
3. Création de la BDD sur MySQL	8
4. Ajout du stockage des images dans la BDD	10
4.1. Création de la table	10
4.2. Ajout de données dans la table	10
5. Réalisation des requêtes SQL	12
5.1. Récupérer toutes les plantes d'un groupe spécifique	12
5.2. Afficher toutes les alertes liées à un bac particulier	12
5.3. Voir la consommation d'énergie totale des bacs	12
5.4. Trouver tous les capteurs installés sur un bac	12
5.5. Afficher les informations sur les périodes de culture	12
5.6. Récupérer toutes les informations de production d'énergie	12
5.7. Lister les différents types de capteurs et leur date d'installation	13
5.8. Récupérer l'arrosage quotidien pour une recette donnée	13
5.9. Liste de la production d'énergie sur les dernières 24 heures	13
6. Conclusion	14
II. API Météo	15
1. Présentation d'OpenWeatherMap	15
2. Intégration de l'API	15
III. Construction du site Web	16
IV. Sécurité du site Web	23
1. Stockage des informations	23
2. Connexion au site	24
3. Création d'un compte utilisateur	26
4. L'onglet Profil	27
5. Changer de mot de passe	28

6.	Déconnexion	28
7.	Protections sur le LattePanda	28
V.	Styles Alternatifs	29
1.	Daltonismes	30
2.	Contraste élevé	30
3.	Mode nuit ou sombre	30
VI.	Normes RGG/RGAA.....	31
VII.	Conclusion.....	34

Introduction

Ce travail de groupe se base sur l'exploitation du projet Opti'Plant mené par les étudiants du parcours All. L'objectif est donc de comprendre les travaux qui ont été menés par ces derniers, et d'étudier le fonctionnement du système actuel afin de pouvoir y apporter de nouveaux éléments.

Le projet Opti'Plant est un système dont la tâche est de permettre de faire pousser divers types de semis dans des bacs au sein de l'IUT. Le système se compose donc de 4 bacs équipés de capteurs d'humidité permettant d'accueillir chacun des plants d'une espèce. Un système d'irrigation automatique utilisant de l'eau de pluie permet de réguler l'humidité de chacun des bacs pour garantir un apport idéal aux plantes en développement. Enfin, des ombrières permettent de contrôler l'exposition au soleil de chaque bac. Tout ce système est alors contrôlé par un réseau de modules d'automates mis en place par les étudiants du parcours All.

La tâche des étudiants du parcours ESE est donc de mettre en place une interface Web qui permette d'obtenir les informations de fonctionnement du projet depuis une connexion à un site hébergé au sein de l'IUT.

I. Partie BDD et requêtes SQL

1. Introduction

Dans le cadre du projet, il a été demandé de concevoir et de mettre en œuvre une base de données (BDD) pour le système automatisé Opti'Plant, un système hydroponique intelligent. Ce système vise à optimiser la gestion des ressources, notamment l'eau, les nutriments et l'énergie, afin d'assurer une croissance optimale des plantes dans un environnement contrôlé. La mission principale a consisté à créer une base de données robuste et fonctionnelle, capable de stocker et de gérer les données essentielles au bon fonctionnement du système, ainsi qu'à fournir des requêtes utiles pour l'extraction et l'analyse de ces données.

La base de données joue un rôle central dans le système Opti'Plant, car elle permet de stocker des informations telles que les paramètres environnementaux (température, humidité, pH, etc.), les données des capteurs, les recettes d'irrigation, les informations sur les plantes, et bien d'autres. Ces données sont ensuite utilisées par les différents modules du système (irrigation, ombrière, production d'énergie, etc.) pour prendre des décisions automatisées et optimiser les processus.

Le travail s'est articulé autour de plusieurs étapes clés :

- 1) **Conception de la base de données** : La structure de la base de données a été définie en identifiant les entités, les relations et les attributs nécessaires pour répondre aux besoins du système. Cette étape a été réalisée à l'aide d'un Modèle Conceptuel de Données (MCD) et d'un Modèle Logique de Données (MLD), qui ont permis de formaliser les relations entre les différentes tables et de garantir une structure cohérente et optimisée.
- 2) **Implémentation de la base de données** : La base de données a été créée en utilisant un système de gestion de base de données (SGBD) adapté, en veillant à ce qu'elle soit normalisée et optimisée pour les performances. La base de données a été implémentée en SQL.
- 3) **Ajout de données de simulation** : Afin de tester la base de données et de valider son fonctionnement, des données de simulation représentatives des conditions réelles d'utilisation du système ont été générées et insérées. Ces données incluent des informations sur les plantes, les bacs, les capteurs, les alertes et les recettes d'irrigation.
- 4) **Création de requêtes utiles** : Un ensemble de requêtes SQL a été élaboré pour permettre l'extraction d'informations spécifiques de la base de données, telles que les données des capteurs, les historiques d'irrigation, ou encore les besoins en nutriments des plantes. Ces requêtes sont essentielles pour le suivi et l'analyse des performances du système.

En somme, cette partie présente en détail les différentes étapes du travail, depuis la conception de la base de données jusqu'à la validation de son fonctionnement à travers des requêtes et des tests. Il explique également comment cette base de données s'intègre dans l'écosystème global du système Opti'Plant, en fournissant des données essentielles pour le contrôle et l'optimisation des processus automatisés.

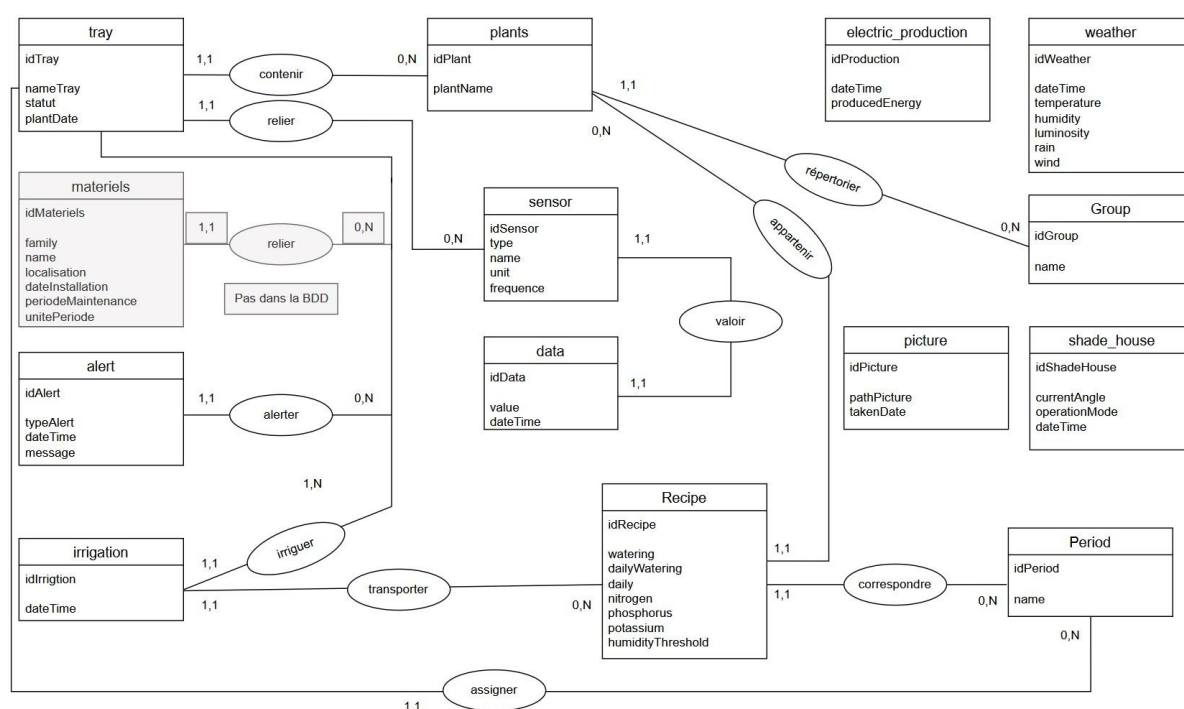
2. Réalisation du MCD/MLD

Afin de réaliser la base de données sous MySQL, il est essentiel de visualiser la structure des données à l'aide de schémas conceptuels et logiques. Ces schémas permettent de comprendre l'organisation des données, les relations entre les différentes entités, et de garantir une conception cohérente et optimale de la base de données.

2.1. Modèle Conceptuel de Données (MCD)

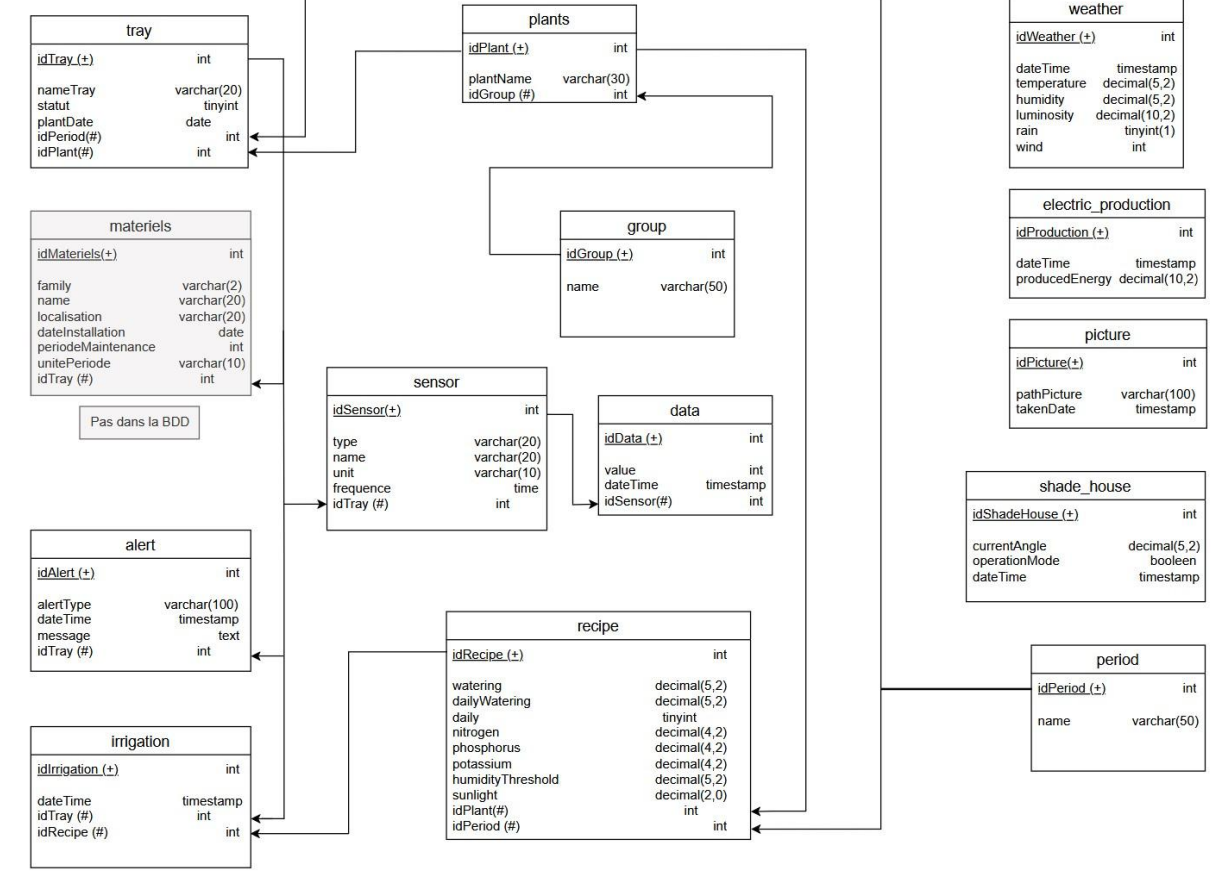
Le Modèle Conceptuel de Données (MCD) représente la structure globale des données du système Opti'Plant. Il décrit les entités, leurs attributs et les relations entre elles. Ce modèle permet de visualiser l'organisation des données et de comprendre comment les différentes parties du système interagissent.

Voici les principales entités du MCD :



Le Modèle Logique de Données (MLD) est une traduction du MCD en un modèle

Voici les principales tables du M.I.D. :

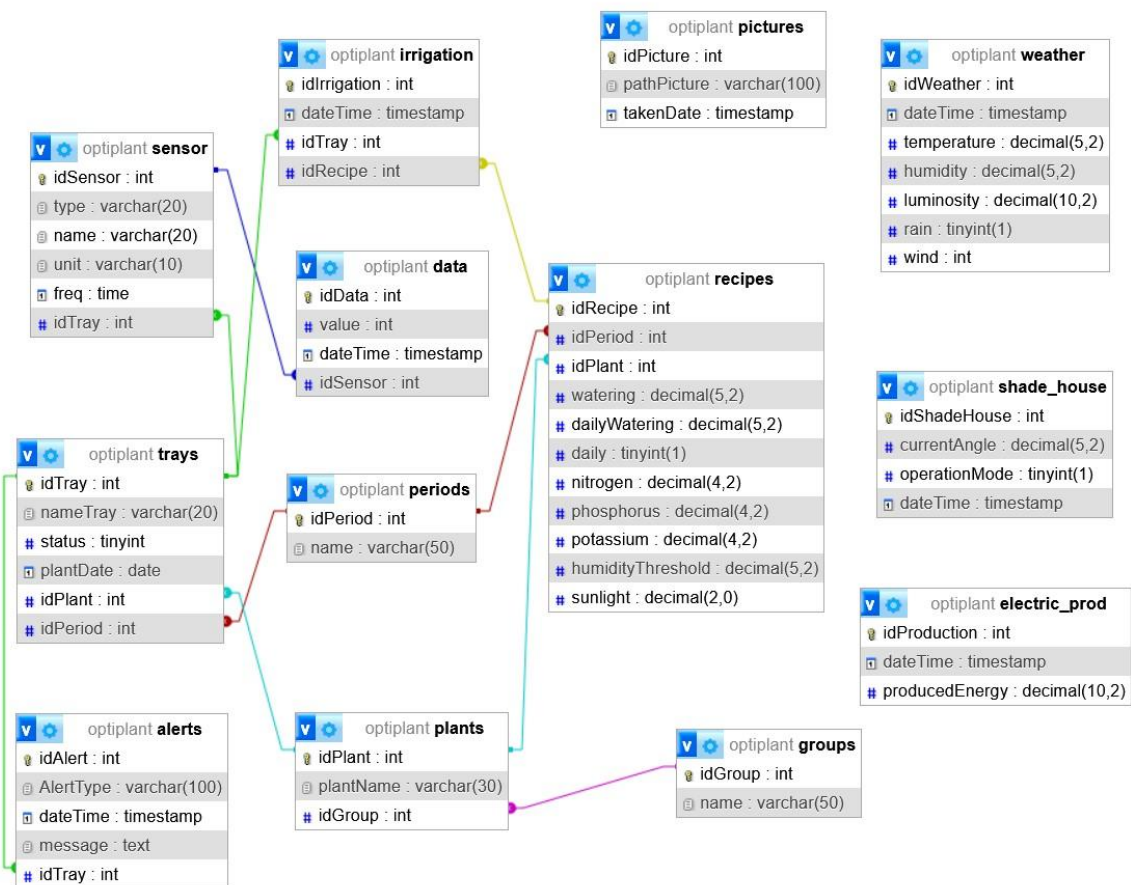


Par exemple :

- Un bac est lié à une période de croissance et à une plante.
- Un capteur est lié à un bac, et les données collectées par ce capteur sont liées au capteur.
- Une recette d'irrigation est liée à une plante et à une période de croissance.

3. Création de la BDD sur MySQL

Après avoir réalisé le MCD/MLD de la base de données, la création des tables de la base de données est réalisée sur le logiciel MySQL. Cette étape consiste à traduire le modèle conceptuel en un schéma physique en utilisant des instructions SQL pour définir chaque table et ses attributs.



Après la création des tables, des simulations ont été réalisées en ajoutant des données dans les tables pour vérifier le bon fonctionnement de la base de données. Ces

simulations ont permis de s'assurer que les relations entre les tables étaient correctement établies et que les contraintes d'intégrité étaient respectées. Par exemple, des données ont été insérées dans la table sensor pour simuler des capteurs, et des enregistrements ont été ajoutés à la table data pour simuler les données collectées par ces capteurs. Ces tests ont confirmé que la base de données était opérationnelle et prête à être utilisée pour la gestion des plantes et des capteurs dans le système OptiPlant.

Voici des exemples de commande SQL utilisée pour remplir les tables avec des données de simulation. Ces commandes illustrent comment les données peuvent être insérées dans chaque table pour tester le bon fonctionnement de la base de données :

```
1 INSERT INTO periods (idPeriod, periodName) VALUES
2 (1, 'Semis'),
3 (2, 'Developpement des racines'),
4 (3, 'Croissance végétative'),
5 (4, 'Floraison et fructification');
```

```
1 INSERT INTO recipes (idRecipe, idPeriod, idPlant, watering, dailyWatering, daily, nitrogen, phosphorus, potassium, humidityThreshold) VALUES
2 (1, 1, 1, 1.00, 3.00, 1, 5.33, 5.33, 5.33, 50.00),
3 (2, 2, 1, 12.00, 1.00, 1, 72.00, 24.00, 24.00, 40.00),
4 (3, 1, 3, 2.00, 3.00, 1, 6.67, 6.67, 6.67, 50.00),
5 (4, 2, 3, 16.00, 1.00, 1, 64.00, 32.00, 64.00, 40.00),
6 (5, 3, 3, 16.00, 0.50, 0, 32.00, 64.00, 96.00, 40.00);
```

Ces commandes SQL permettent de remplir les tables avec des données de test, ce qui est essentiel pour vérifier que les relations entre les tables fonctionnent correctement et que les contraintes d'intégrité sont respectées. Ces simulations aident également à s'assurer que la base de données est prête à être utilisée par le système.

Voici le résultat obtenu :

- Table periods

idPeriod	name
1	Semis
2	Developpement des racines
3	Croissance végétative
4	Floraison et fructification

- Table recipes

idRecipe	idPeriod	idPlant	watering	dailyWatering	daily	nitrogen	phosphorus	potassium	humidityThreshold	sunlight
1	1	1	1.60	3.00	1	5.33	5.33	5.33	50.00	0
2	2	1	12.00	1.00	1	72.00	24.00	24.00	40.00	0
3	1	3	2.00	3.00	1	6.67	6.67	6.67	50.00	0
4	2	3	16.00	1.00	1	64.00	32.00	64.00	40.00	0
5	3	3	16.00	0.50	0	32.00	64.00	96.00	40.00	0

4. Ajout du stockage des images dans la BDD

4.1. Création de la table

Dans le cadre de l'optimisation du système Opti'Plant, il convient de se concentrer sur la table pictures, qui gère les images capturées par la caméra et les stocke dans la base de données à intervalles réguliers. Cette table permet de conserver un historique des images prises par le système, ce qui est essentiel pour le suivi visuel des plantes et l'analyse de leur croissance au fil du temps.

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
<input type="checkbox"/> 1	idPicture	int			Non	Aucun(e)		AUTO_INCREMENT	Modifier Supprimer Plus
<input type="checkbox"/> 2	pathPicture	varchar(100)	utf8mb4_general_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/> 3	takenDate	timestamp			Non	CURRENT_TIMESTAMP		DEFAULT_GENERATED	Modifier Supprimer Plus

Tout d'abord, la table pictures est créée sur MySQL avec les colonnes suivantes :

- **idPicture** : De type int, cette colonne sert d'identifiant unique pour chaque image. Elle est définie comme une clé primaire et est auto-incrémentée, ce qui garantit que chaque image ajoutée à la base de données possède un identifiant distinct.
- **pathPicture** : De type varchar(100), cette colonne stocke le chemin d'accès ou l'URL de l'image. Le choix d'un varchar avec une limite de 100 caractères permet de stocker des chemins de fichiers ou des URLs tout en limitant l'espace utilisé dans la base de données.
- **takenDate** : De type timestamp, cette colonne enregistre la date et l'heure exactes auxquelles l'image a été capturée. La valeur par défaut est définie sur CURRENT_TIMESTAMP, ce qui permet d'automatiser l'enregistrement du moment où l'image est ajoutée à la base de données.

Ces choix de conception permettent de gérer efficacement les images capturées par le système, en assurant une traçabilité précise et une organisation optimale des données visuelles. La structure de la table pictures est conçue pour être simple, tout en répondant aux besoins de stockage et de récupération des images dans le cadre du système Opti'Plant.

4.2. Ajout de données dans la table

Ensuite, un programme Python a été développé pour permettre l'ajout des images capturées par la caméra dans la base de données. Ce programme utilise la bibliothèque

mysql.connector pour se connecter à la base de données MySQL et insérer les données dans la table pictures. Voici les principales fonctionnalités du programme :

- **Connexion à la base de données** : Le programme établit une connexion à la base de données en utilisant les informations de configuration (hôte, utilisateur, mot de passe, base de données) fournies dans un fichier JSON. Cette connexion est gérée de manière sécurisée pour éviter les fuites de ressources.
- **Capture d'images** : Le programme capture des images à intervalles réguliers (définis par la variable image_period) à partir d'un flux vidéo généré par FFmpeg. Chaque image est sauvegardée dans un répertoire spécifique avec un nom de fichier basé sur un timestamp pour garantir l'unicité.
- **Insertion des images dans la base de données** : Après chaque capture, le programme insère le chemin de l'image (pathPicture) et la date de capture (takenDate) dans la table pictures de la base de données. La requête SQL utilisée pour l'insertion est la suivante :

```
INSERT INTO Pictures (pathPicture, takenDate) VALUES (%s, %s)
```

Les valeurs sont passées sous forme de paramètres pour éviter les injections SQL et garantir la sécurité des données.

- **Gestion des erreurs** : Le programme inclut une gestion des erreurs pour capturer et afficher les éventuelles erreurs lors de la connexion à la base de données ou de l'exécution des requêtes SQL. Cela permet de diagnostiquer rapidement les problèmes et d'assurer la robustesse du système.
- **Threading et gestion des processus** : Le programme utilise le module threading pour exécuter la capture d'images en arrière-plan, tout en maintenant un serveur Flask pour diffuser le flux vidéo en direct. Cela permet une exécution simultanée et efficace des différentes tâches.
- **Arrêt propre du programme** : Le programme est conçu pour gérer les interruptions (comme un CTRL+C) en arrêtant proprement le processus FFmpeg et en fermant la connexion à la base de données avant de quitter.

Ce programme permet ainsi d'automatiser l'enregistrement des images dans la base de données, tout en assurant une gestion robuste des ressources et des erreurs. Il s'intègre parfaitement dans le système Opti'Plant pour fournir un suivi visuel continu des plantes.

5. Réalisation des requêtes SQL

Cette partie concerne l'élaboration de requêtes SQL permettant de récupérer des informations de manière rapide et efficace. Quelques exemples utiles sont présentés ici, mais une liste complète des requêtes est disponible en annexe. Ces requêtes sont adaptées aux besoins réels du système Opti'Plant et optimisées pour une utilisation pratique.

5.1. Récupérer toutes les plantes d'un groupe spécifique

```
SELECT * FROM `plants` WHERE `idGroup` = 1;
```

Permet d'afficher toutes les plantes appartenant à un groupe spécifique (par exemple, les plantes à feuilles).

5.2. Afficher toutes les alertes liées à un bac particulier

```
SELECT * FROM `alerts` WHERE `idTray` = 1;
```

Permet de visualiser toutes les alertes générées pour un bac spécifique.

5.3. Voir la consommation d'énergie totale des bacs

```
SELECT `nameTray`, SUM(`producedEnergy`) AS `consommation_totale`  
FROM `electric_prod`  
JOIN `trays` ON `electric_prod`.`idTray` = `trays`.`idTray`  
GROUP BY `nameTray`;
```

Calcule la consommation d'énergie totale pour chaque bac.

5.4. Trouver tous les capteurs installés sur un bac

```
SELECT * FROM `sensor` WHERE `idTray` = 1;
```

Liste tous les capteurs associés à un bac spécifique.

5.5. Afficher les informations sur les périodes de culture

```
SELECT * FROM `periods`;
```

5.6. Récupérer toutes les informations de production d'énergie

```
SELECT * FROM `electric_prod`;
```

Affiche toutes les données de production d'énergie.

5.7. Lister les différents types de capteurs et leur date d'installation

```
SELECT `type`, `dateInstallation` FROM `sensor`;
```

Affiche les types de capteurs et leur date d'installation.

5.8. Récupérer l'arrosage quotidien pour une recette donnée

```
SELECT r.`idGroup`, r.`watering`, r.`dailyWatering`  
FROM `recipes` r  
JOIN `groups` g ON r.`idGroup` = g.`idGroup`;
```

Affiche la quantité d'eau quotidienne pour une recette spécifique.

5.9. Liste de la production d'énergie sur les dernières 24 heures

```
SELECT * FROM `electric_prod` WHERE `dateTime` >= NOW() - INTERVAL 1 DAY;
```

Affiche la production d'énergie au cours des dernières 24 heures.

6. Conclusion

La partie du projet d'implémentation de la base de données pour le système automatisé Opti'Plant a abouti à une structure robuste et fonctionnelle, essentielle pour gérer les données clés du système hydroponique intelligent. En suivant une approche méthodique, nous avons transformé les besoins en une base de données optimisée pour la gestion des ressources comme l'eau, les nutriments et l'énergie.

La création du Modèle Conceptuel de Données (MCD) et du Modèle Logique de Données (MLD) a permis de structurer les données de manière cohérente, en identifiant les entités, les attributs et les relations entre les tables. Cette étape a été cruciale pour garantir une base de données normalisée et performante. L'implémentation sous MySQL a traduit ces modèles en une structure exploitable, avec des tables bien définies et des relations claires.

Les simulations avec des données de test ont validé le bon fonctionnement de la base de données, confirmant que les relations et les contraintes d'intégrité sont respectées. Les requêtes SQL développées permettent d'extraire des informations essentielles, comme les données des capteurs, les historiques d'irrigation et les besoins en nutriments, facilitant ainsi le suivi et l'analyse des performances du système.

L'intégration de la gestion des images via la table pictures a enrichi le système en offrant un suivi visuel des plantes. Le programme Python pour l'insertion automatique des images a démontré la flexibilité et l'évolutivité du système.

En résumé, cette partie a abouti à une base de données solide et adaptée aux besoins d'Opti'Plant, prête pour une utilisation en production. Elle constitue une pierre angulaire pour le développement futur du système, en permettant une prise de décision automatisée et une optimisation des processus.

II. API Météo

1. Présentation d'OpenWeatherMap

OpenWeatherMap est une plateforme météorologique globale offrant des données en temps réel, des prévisions horaires, quotidiennes et historiques via des API RESTful. Elle se distingue par :

- **Couverture mondiale** : Données provenant de plus de 200 000 stations météo, satellites et radars.
- **Polyvalence** : Accès à des données comme la température, l'humidité, la vitesse du vent, les précipitations et l'UV.
- **Documentation claire** : Des endpoints bien structurés (e.g., /weather, /forecast) et des bibliothèques SDK pour Python, JavaScript, etc.
- **Gratuité partielle** : Un plan gratuit avec 1 000 appels API/jour, idéal pour des projets tels que celui-ci.

2. Intégration de l'API

Pour pouvoir utiliser les données des stations météo de OpenWeatherMap, il est nécessaire d'obtenir une clé d'authentification, ou Token. Ce Token doit être inséré dans le programme qui réalise les requêtes vers le site de OpenWeatherMap afin que l'accès lui soit autorisé.

Pour générer ce Token, il faut donc créer un compte sur le site en ligne et choisir un des abonnements. Dans le cas du projet Opti'Plant, la version gratuite avec les 1000 requêtes journalières est largement suffisante et il n'y a donc pas besoin de souscrire à un abonnement payant.

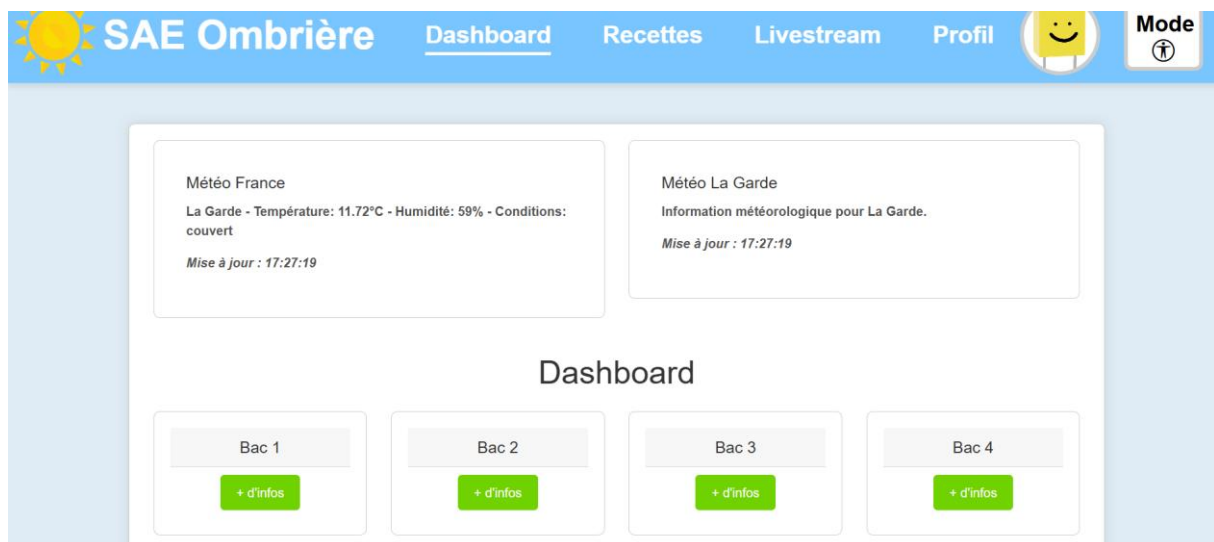
Un code assez simple permet alors d'effectuer des requêtes de façon automatique dès qu'un utilisateur a besoin d'afficher les données météo de La Garde. Ces données sont transmises par OpenWeatherMap sous forme d'un fichier JSON qu'il est possible de modifier pour ne garder que les données utiles dans le cadre du projet.

Une autre fonctionnalité de cette API est de pouvoir faire une requête de prévision météo. En effet, il est possible de demander une estimation sur les 5 prochains jours et ce toutes les 3 heures. Cela est donc pratique si l'on souhaite pouvoir anticiper des changements de météo et donc adapter l'irrigation et l'exposition au soleil en amont, permettant d'optimiser l'utilisation des ressources et la pousse des plantes.

III. Construction du site Web

Concernant la partie interface web, cette dernière est décomposée en 3 pages différentes. La page principale, le Dashboard, la page des recettes et la page des bacs, qui affiche uniquement les données du bac sur lequel on a cliqué. Pour réaliser ce site, 4 langages de programmation ont été utilisés, le PHP qui permet de faire des requêtes de données à la base de données stockées sur PHPMyAdmin. Le HTML qui affiche toutes les données récupérées grâce au PHP. Le CSS qui est un éditeur de style, affiche les informations écrites avec le HTML à des endroits précis tout en ajoutant de la couleur. Et enfin le JavaScript, qui permet de rendre la page dynamique, afin d'avoir des menus déroulants ou bien des animations. Concernant le Dashboard, il est composé des éléments suivants :

- Les informations météorologiques
- Le numéro de chaque bac, avec la possibilité d'aller sur la page du bac sélectionné
- La possibilité d'aller sur la page de la liste des recettes
- Un tableau avec la liste des 5 dernières alertes recensées chronologiquement



Dashboard



Concernant les accès aux bacs ainsi qu'à la liste des recettes, le procédé est simplement de créer un bouton cliquable qui redirige vers le lien de la page concernée avec les balises en HTML : ` « Texte du bouton » `

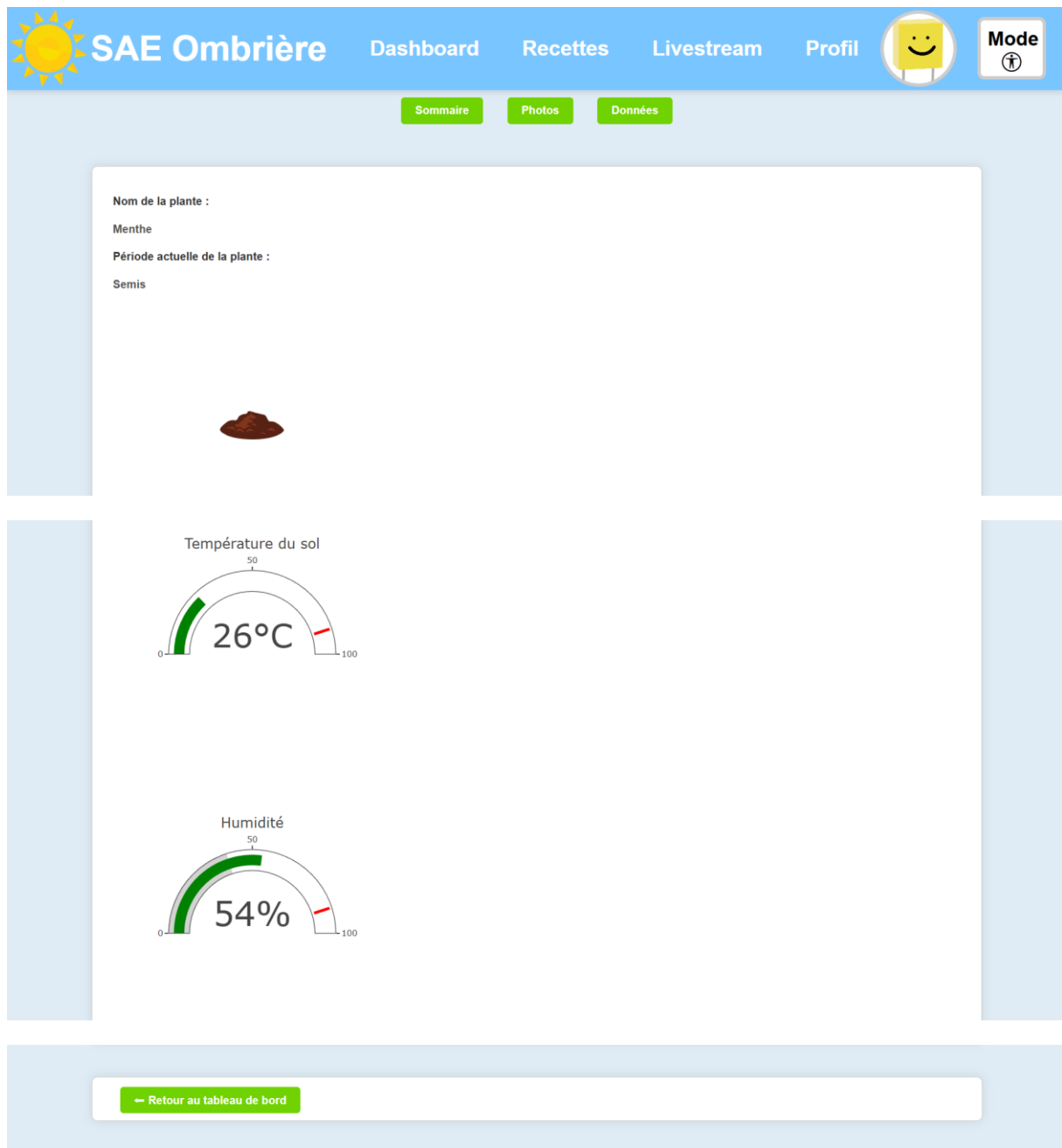
Pour la partie des alertes, il a fallu faire une requête SQL en PHP afin de récupérer les données nécessaires pour l'affichage chronologique des 5 dernières alertes. Pour ce faire, voici la requête :

```
// Requête SQL pour récupérer les 5 dernières alertes
$sql = "SELECT message, dateTime FROM alerts ORDER BY dateTime DESC LIMIT 5";
$stmt = $pdo_optiplant->prepare($sql);
$stmt->execute();
// Récupérer les alertes
$alerts = $stmt->fetchAll( mode: PDO::FETCH_ASSOC);
```

On demande à la BDD de sélectionner le message ainsi que la date de la BDD alerts, de la plus récente à la plus ancienne (ORDER BY dateTime DESC) et que seules les 5 dernières alertes soient récupérées (LIMIT 5). Ainsi, la page Dashboard est correctement configurée avec la partie HTML qui affiche les données et le JavaScript qui s'occupe de rendre les pages dynamiques.

Concernant les autres pages web, étant donné que le groupe était composé de 3 personnes et que le temps nous était limité, chaque personne a travaillé sur des parties différentes. J'ai personnellement travaillé principalement sur les bacs (dans les données de chaque bac, et un petit peu le sommaire) ainsi que sur le Dashboard.

Concernant les bacs, je vais d'abord parler du sommaire. Ce dernier affiche les informations simples concernant la plante qui se trouve dedans, ainsi que son état, un graphique sur la température du sol et l'humidité.



```

    erDiagram
        tray ||--o{ plants : "has"
        tray ||--o{ materials : "has"
        tray ||--o{ sensor : "has"
        tray ||--o{ irrigation : "has"
        tray ||--o{ alert : "has"
        plants ||--o{ group : "belongs to"
        plants ||--o{ data : "has"
        plants ||--o{ recipe : "has"
        weather ||--o{ electric_production : "has"
        weather ||--o{ picture : "has"
        weather ||--o{ shade_house : "has"
        weather ||--o{ period : "has"
        materials ||--o{ sensor : "has"
        materials ||--o{ data : "has"
        materials ||--o{ recipe : "has"
        sensor ||--o{ data : "has"
        sensor ||--o{ recipe : "has"
        data ||--o{ recipe : "has"
        recipe ||--o{ irrigation : "has"
        recipe ||--o{ alert : "has"
        recipe ||--o{ period : "has"
  
```

The diagram illustrates the database structure for a smart garden system. It includes tables for various components and their relationships:

- tray**: Contains information about the tray, including its ID, name, status, plant date, and associated plant and materials.
- plants**: Contains information about the plants, including their ID, name, and associated group and data.
- weather**: Contains weather-related information, including date, time, temperature, humidity, luminosity, rain, and wind.
- electric_production**: Contains information about electric production, including date, time, and produced energy.
- picture**: Contains information about pictures, including the picture ID, path, and taken date.
- shade_house**: Contains information about the shade house, including the shade house ID, current angle, operation mode, and date.
- period**: Contains information about the period, including the period ID and name.
- materials**: Contains information about materials, including their ID, family, name, location, installation date, maintenance period, and associated tray, sensor, and data.
- sensor**: Contains information about sensors, including their ID, type, name, unit, frequency, and associated tray, data, and recipe.
- data**: Contains information about data, including the data ID, value, date, time, and associated sensor and recipe.
- group**: Contains information about groups, including the group ID and name.
- recipe**: Contains information about recipes, including the recipe ID, watering, daily watering, daily nitrogen, phosphorus, potassium, humidity threshold, sunlight, and associated plant, period, and irrigation.
- irrigation**: Contains information about irrigation, including the irrigation ID, date, time, and associated tray and recipe.
- alert**: Contains information about alerts, including the alert ID, alert type, date, time, message, and associated tray.

Ce travail fut simplifié mais le chemin était tout de même très long. Le point de départ est la BDD plants. Afin de relier la plante au bac, rien de plus simple, faire le lien entre plants et tray via idPlant. Pour la période, il faut savoir que chaque recette d'irrigation est envoyée en fonction de la plante ainsi que de son état. Pour cela, en partant de plants, il faut se rendre à recipe (via idPlant) puis aller à period (via idPeriod qui se trouve dans recipe). Voici donc à quoi ressemble la requête pour afficher le nom de la plante ainsi que sa période :

```
// Requête SQL pour récupérer les informations du bac
$sql = "SELECT trays.*,
             plants.plantName AS plantName,
             periods.name AS periodName
        FROM trays
        LEFT JOIN plants ON trays.idPlant = plants.idPlant
        LEFT JOIN recipes ON plants.idPlant = recipes.idPlant
        LEFT JOIN periods ON recipes.idPeriod = periods.idPeriod
        WHERE trays.idTray = :idTray";
$stmt = $pdo_optiplant->prepare($sql);
$stmt->bindParam( param: ':idTray', &var: $idTray, type: PDO::PARAM_INT);
$stmt->execute();
$bac = $stmt->fetch( mode: PDO::FETCH_ASSOC);

// Si aucun bac n'est trouvé, afficher un message
if (!$bac) {
    die('<div class="alert alert-warning">Aucune information disponible pour ce bac.</div>');
}
```

Passons maintenant à la partie des données, voici à quoi ressemble cet onglet :

Données d'Irrigation

Date et Heure	Recette	Âge (en heures)
Aucune donnée trouvée pour les dernières 24 heures		

Données des Capteurs

Type	Valeur	Unité	Fréquence
Humidité	54	%	00:10:43 secondes
Luxmètre	N/A	Lux	03:05:04 secondes
Température	26	°C	03:12:00 secondes

Dernières Alertes

- Température élevée (28/02/2025 13:42)
- Humidité basse (28/02/2025 13:42)

Retour au tableau de bord

On voit donc 3 tableaux qui en ressortent concernant les données d'irrigation, les données des capteurs ainsi que les 3 dernières alertes liées à ce bac. S'il y a plus de 3 alertes concernant ce bac, toutes les alertes recensées peuvent être affichées en appuyant sur le bouton « Voir tout » qui s'affiche uniquement s'il y a plus de 3 alertes sur ce bac. Les alertes sont limitées aux 5 dernières une fois l'appui sur le bouton, afin de ne pas dresser une liste déroulante trop longue.

Concernant les requêtes SQL pour les données d'irrigation, des capteurs ainsi que des alertes, ces dernières sont simples.

```
// Requête pour récupérer les données d'irrigation des dernières 24 heures
$sqlIrrigation = "SELECT *, TIMESTAMPDIFF(HOUR, dateTime, NOW()) AS hoursAgo
FROM irrigation
WHERE idTray = :idTray AND dateTime >= NOW() - INTERVAL 24 HOUR
ORDER BY dateTime DESC";
$stmtIrrigation = $pdo_optiplant->prepare($sqlIrrigation);
$stmtIrrigation->bindParam( param: ':idTray', &var: $idTray, type: PDO::PARAM_INT);
$stmtIrrigation->execute();
$irrigations = $stmtIrrigation->fetchAll( mode: PDO::FETCH_ASSOC);
```

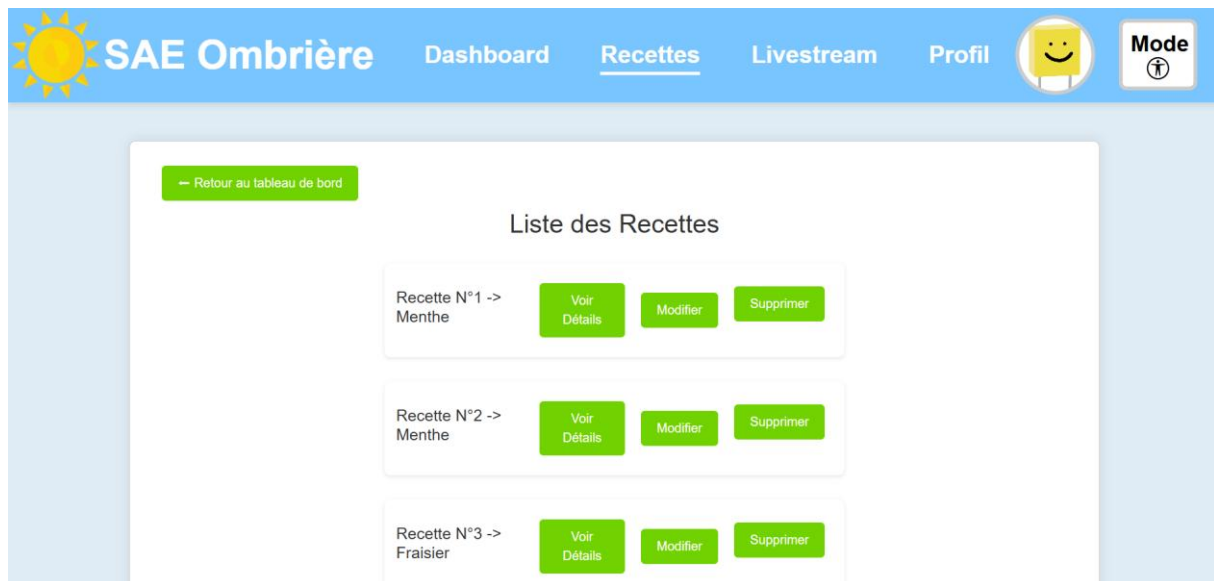
```
// Requête SQL pour récupérer les alertes spécifiques au bac
$sql = "SELECT message, dateTime FROM alerts WHERE idTray = :idTray ORDER BY dateTime DESC LIMIT 5";
$stmt = $pdo_optiplant->prepare($sql);
$stmt->bindParam( param: ':idTray', &var: $idTray, type: PDO::PARAM_INT);
$stmt->execute();
$alerts = $stmt->fetchAll( mode: PDO::FETCH_ASSOC);
```

```
// Requête SQL pour récupérer les informations des capteurs avec leurs dernières valeurs
$sqlSensorsWithData = "SELECT sensor.idSensor, sensor.type, sensor.unit, sensor.freq, data.value
FROM sensor
LEFT JOIN data ON sensor.idSensor = data.idSensor
WHERE sensor.idTray = :idTray
ORDER BY sensor.type ASC";

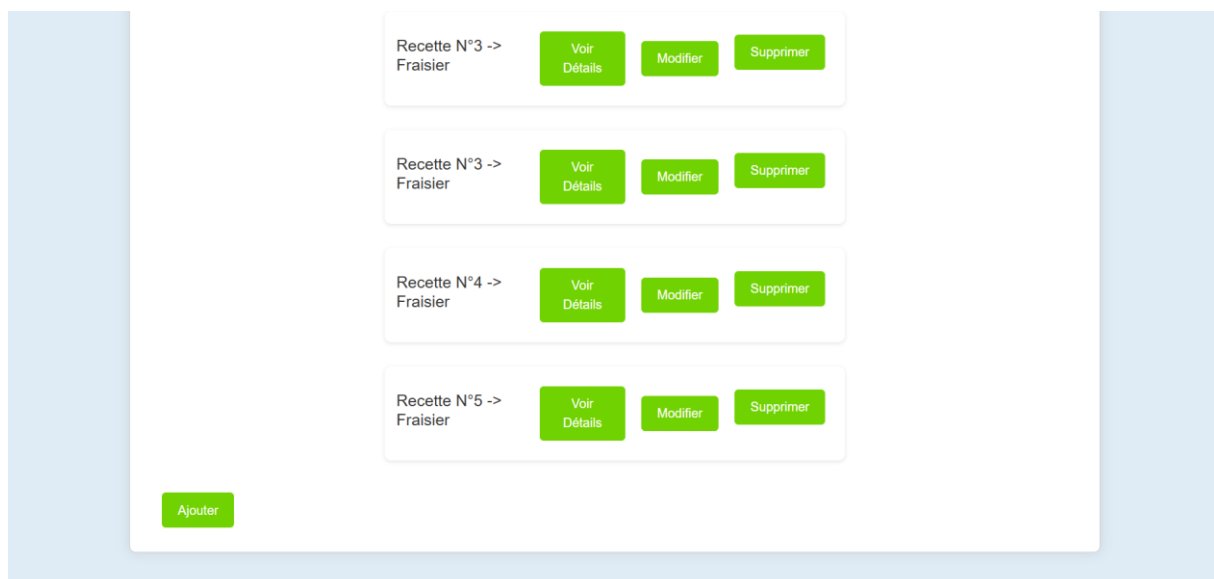
$stmtSensorsWithData = $pdo_optiplant->prepare($sqlSensorsWithData);
$stmtSensorsWithData->bindParam( param: ':idTray', &var: $idTray, type: PDO::PARAM_INT);
$stmtSensorsWithData->execute();
$sensorsWithData = $stmtSensorsWithData->fetchAll( mode: PDO::FETCH_ASSOC);
```

Puis, grâce au code HTML, les données sont affichées dans des tableaux.

Concernant la page des recettes, cette dernière se présente comme suit :



Cette page affiche toutes les données des différentes recettes qui peuvent être irriguées. On remarque que ces recettes peuvent être observées si l'on appuie sur voir détails afin de voir toutes les informations concernant la recette sélectionnée. Il est également possible de la modifier ou de la supprimer.



En bas de la page, il y a également la possibilité d'ajouter de nouvelles recettes dans la BDD si de nouvelles plantes sont ajoutées dans les bacs ou bien qu'une nouvelle recette veuille être irriguée.

IV. Sécurité du site Web

L'objectif du travail mené sur le projet Opti'Plant étant de mettre en place un système d'interface utilisateur accessible à distance, il faut alors faire en sorte de garantir un certain niveau de sécurité et de protection des données, surtout en ce qui concerne le livestream de la caméra des bacs.

Cette sécurisation du site se fera donc dans deux registres différents :

- Tout d'abord, la mise en place d'un système de comptes utilisateurs et de droits.
- Puis, par l'ajout de sécurité d'accès sur le LattePanda.

La première étape dans la sécurisation du site est de ne pas permettre à toutes les personnes disposant d'une connexion vers le LattePanda de voir ce qu'il se passe sur l'interface Web.

Pour ce faire, il est possible d'utiliser un système de session et de comptes utilisateurs avec des mots de passe. Cette solution permet aussi de contrôler l'accès aux différentes rubriques de l'interface pour les personnes qui utilisent le projet.

1. Stockage des informations

Pour qu'un compte puisse être reconnu et démarrer une session sur le site, il faut d'abord que celui-ci soit stocké quelque part. Travaillant déjà avec les outils MySQL et PHPMyAdmin, il est assez facile de créer une nouvelle base de données contenant l'ensemble des informations importantes pour le système de login et de session.

Cette base se structure donc de la manière suivante :

id	username	password	email	lerole	profile_photo	last_login	mode
9	othmane83	\$2y\$10\$rzUHbgGvKXtDbvQE0k3juy7LutU57kdJhUQYhrA1pV...	elbertalothmane0@gmail.com	admin	images/nyquit7.jpg	2025-02-28 14:46:08	contraste élevé
12	Lilian	\$2y\$10\$P/rMzUK69FOEWzv3w/K1uVmAy8n7Ff5qURjkb.avrd...	liliancarriere@gmail.com	admin	images/nyquit6.jpg	2025-03-10 23:04:47	tritanopie
13	Jean Damien Damien	\$2y\$10\$yPEHo6uxLImYUj09AEKTu9C1hkpJy3IIW5PYUxu5Ze...	JDD.test@gmail.com	membre	images/nyquit1.jpg	NULL	default
14	Pascal Obispo	\$2y\$10\$WopRLkd29mQMIVxfVhFgdO5v/FJA4j0w9eKeY884sbj...	Obispo.Pacal@star.net	membre	images/nyquit1.jpg	NULL	default

Extrait de la table users

Explications des différentes rubriques :

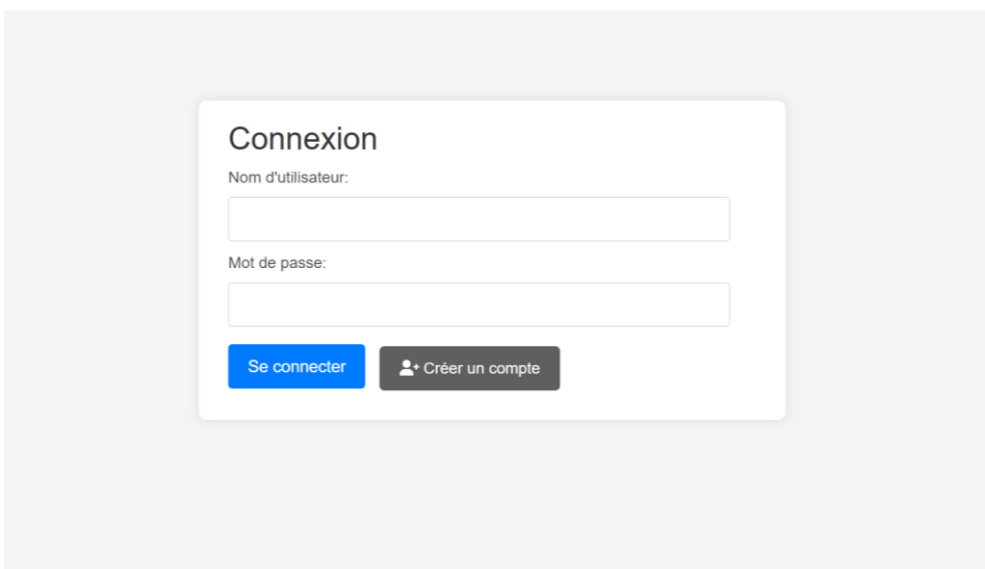
- id. Il s'agit d'un chiffre qui s'incrémente automatiquement et qui est unique pour chaque utilisateur.
- username. L'utilisateur peut choisir un alias avec lequel les requêtes seront alors effectuées
- password. Le mot de passe que l'utilisateur choisi est hashé afin qu'il ne soit pas écrit de façon explicite dans la base de données.
- email. L'utilisateur fournit une adresse avec laquelle il serait possible de le contacter si besoin.

- `lerole`. Cette rubrique permet de définir les informations qui seront visibles pour l'utilisateur en fonction du niveau d'accès qu'il lui est attribué. Par défaut, le rôle attribué après la création d'un nouveau compte est celui de membre.
- `profile_photo`. Cette zone permet de définir la photo de profil que l'utilisateur choisit parmi une sélection d'image.
- `last_login`. Il est possible de connaître la date de dernière connexion d'un utilisateur sur l'interface Web.
- `mode`. C'est ici qu'est indiqué le mode d'affichage que souhaite utiliser l'utilisateur en fonction de ses préférences ou de problèmes liés à la vision des couleurs.

Cette table « users » est contenue dans une base de données « compte_utilisateur » en parallèle de la base de données « optiplant » sur le LattePanda.

2. Connexion au site

Lorsqu'une personne tente d'accéder au site, elle sera alors dirigée vers la page de connexion qui est la suivante :



The image shows a web form titled "Connexion". It contains two input fields: "Nom d'utilisateur:" and "Mot de passe:". Below the fields are two buttons: a blue button labeled "Se connecter" and a grey button labeled "Créer un compte" with a user icon.

Page de connexion (index.php)

Cette page est similaire à celle que l'on peut rencontrer sur la plupart des autres sites en ligne. L'utilisateur peut soit se connecter à un compte dont il connaît le nom d'utilisateur et le mot de passe, ou alors, cliquer sur un bouton qui le redirigera vers la page de création d'utilisateur.

Cette page ne présente alors aucune indication en lien avec le projet Opti'Plant et l'université.

Le fichier php fonctionne de la façon suivante :

Tout d'abord, le fichier démarre une nouvelle session sur le navigateur de l'utilisateur. L'utilisation de cette session permet notamment de permettre à un utilisateur qui s'est déjà connecté au cours d'une session (c'est-à-dire sans avoir fermé son navigateur ou effacé ses données de navigation) de ne pas avoir à repasser par la page de connexion s'il n'a pas appuyé sur « Se déconnecter ». Cette session est cependant détruite dès lors que l'utilisateur ferme complètement son navigateur Web ou s'il passe par le fichier logout.php normalement attaché au bouton « Se déconnecter » du profil.

Une fois que la session est démarrée, et si l'utilisateur n'a pas déjà une autre session de lancée, alors le programme se met à attendre que la personne appuie sur l'un des boutons pour pouvoir lancer les actions correspondantes.

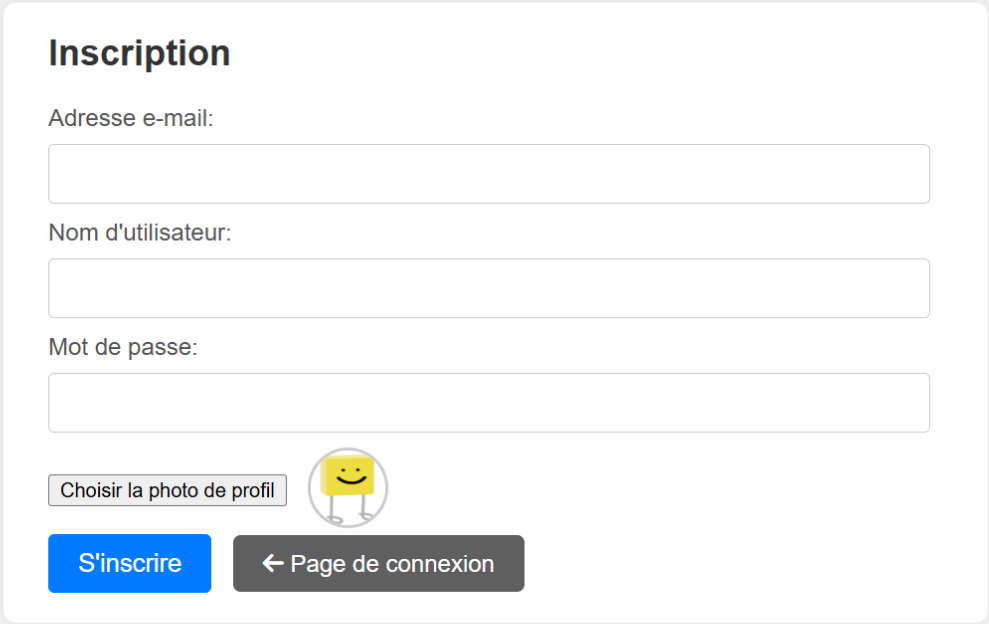
Lorsque l'utilisateur appuie sur le bouton « Se connecter », le programme récupère alors les informations qu'il a entrées et les compare avec celles qui sont disponibles dans la base de données en utilisant des requêtes SQL. S'il s'avère que le programme trouve un compte avec le même nom d'utilisateur, alors il procédera à une vérification des mots de passe hashés. Si les hash correspondent, une nouvelle session est démarrée mais elle est cette fois-ci associée à l'ID du compte utilisé. Le last login est aussi modifié et l'utilisateur est redirigé vers la page dashboard.php.

Au contraire, si le mot de passe ou le nom d'utilisateur ne correspondent pas aux données stockées dans la table users, alors le programme affichera un message d'erreur indiquant que le mot de passe ou le nom d'utilisateur sont incorrects. L'utilisateur peut alors retenter ou décider de se créer un compte.

Lorsque celui-ci décide d'appuyer sur le bouton « Créer un compte », il est alors redirigé vers la page register.php.

3. Création d'un compte utilisateur

La page de création de compte devant laquelle est amené l'utilisateur est la suivante :



Page de création de compte (register.php)

Cette page consiste en un formulaire demandant de renseigner les informations nécessaires à la création d'un compte. Ces informations sont :

- Une adresse e-mail. Elle n'est pas utilisée pour le moment mais pourrait faire l'objet d'ajout de fonctionnalité.
- Un nom d'utilisateur.
- Un mot de passe

Il faut que les informations entrées respectent certaines contraintes pour que le bouton « S'inscrire » puisse fonctionner. En effet, une vérification de la structure de l'adresse e-mail et du mot de passe est effectuée.

Pour l'adresse mail, il faut que celle-ci comporte un « @ » et qu'il soit suivi par quelque chose. L'adresse e-mail n'est pas réellement vérifiée, il s'agit juste de s'assurer qu'elle soit bel et bien sous la forme « ...@... ». Ce qui se trouve à la place des points n'est pas vérifié.

Le nom d'utilisateur n'est soumis à aucune restriction, mais il ne peut pas être le même que celui d'un utilisateur déjà présent dans la base de données. Dans ce cas, un message indiquera que le nom est déjà utilisé et il faut en utiliser un autre.

Le mot de passe est lui aussi soumis à des contraintes qui permettent de renforcer la sécurité. Ces contraintes sont :

Au minimum :

- Une lettre
- Un chiffre
- Une majuscule
- Un caractère spécial

Et il faut que le mot de passe fasse au moins 8 caractères.

En théorie, il est plus difficile pour une personne malveillante de trouver le mot de passe en raison de toutes ces contraintes.

Il est évident qu'aucune alerte n'est levée si le mot de passe existe déjà dans la base de données pour un autre utilisateur.

Il est aussi possible de choisir une photo de profil parmi les 8 images proposées sur le site.

Une fois que toutes les informations ont été renseignées, et que les contraintes sont vérifiées, il est alors possible de cliquer sur le bouton « S'inscrire » et de retourner sur la page de connexion `index.php`.

4. L'onglet Profil

Après s'être connecté ou avoir restauré une session déjà existante, l'utilisateur arrive alors sur la page `dashboard.php`. Sur cette page, il pourra retrouver l'ensemble des informations dont il a besoin et accéder aux différentes pages du projet.

L'utilisateur peut alors cliquer sur sa photo de profil, ou sur le lien profil dans l'entête pour accéder à la page `profil.php`.

La page `profil.php` permet à l'utilisateur de se déconnecter, de changer sa photo de profil et d'accéder à ses informations.

Il peut notamment retrouver son nom d'utilisateur, l'adresse électronique qu'il a renseignée, son mode de couleurs, et le rôle au sein du projet. Il y a aussi un bouton « Changer de mot de passe » qui permet de modifier son mot de passe via la page `changempd.php`.

Pour les personnes ayant le rôle membre, la page est celle décrite au-dessus. Pour les administrateurs, une section administrateur se trouve alors en dessous.

Cette section leur permet de voir l'ensemble des comptes stockés dans la base de données et d'afficher certaines informations les concernant. Les administrateurs peuvent donc savoir à quelle heure s'est connecté un compte pour la dernière fois, voir son rôle, et décider de le promouvoir en tant qu'administrateur si besoin.

5. Changer de mot de passe

La page changempdp.php prend elle aussi la forme d'un formulaire avec trois encarts permettant de réaliser le changement.

La première zone de texte demande l'ancien mot de passe qui sera comparé avec celui de la base de données afin de débloquent l'action du bouton de validation.

Puis, l'utilisateur n'a plus qu'à rentrer le nouveau mot de passe deux fois de façon identique dans les deux zones restantes.

Le nouveau mot de passe est alors hashé et une requête SQL vient modifier la base de données.

6. Déconnexion

Le dernier fichier ayant un rôle dans la sécurité du site est le fichier logout.php. Ce fichier n'a pas de section HTML et il n'est jamais visible depuis le site. Il intervient lorsque l'utilisateur utilise le bouton « Se déconnecter » depuis la page profil.php. Le programme supprime toutes les variables liées à la session en cours, notamment le numéro d'ID associé, et supprime tous les cookies que le site a pu utiliser. Une fois ceci fait, la session est détruite et l'utilisateur est redirigé vers la page index.php.

7. Protections sur le LattePanda

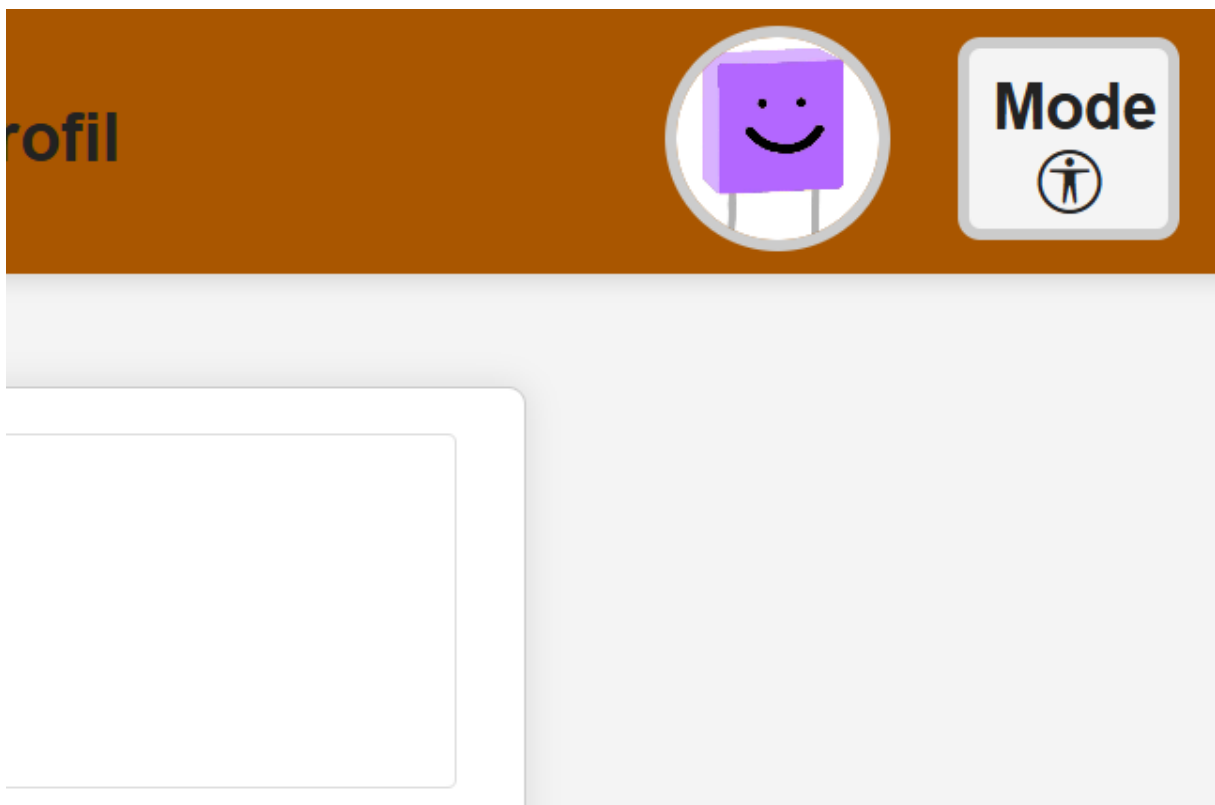
Enfin, si l'on souhaite avoir une sécurité maximale, il faut à la fois prévenir des attaques extérieures, mais aussi rendre plus difficile une attaque depuis l'intérieur. C'est pour cela que certaines restrictions ont alors été installées sur le LattePanda pour tenter de limiter le risque de perturbations du site lors de l'utilisation de ce dernier par une personne ne connaissant pas les procédures.

Pour ce faire, il est possible d'utiliser le terminal de commande Linux et la commande CHMOD, permettant de redéfinir les droits et les accès liés à certains dossiers du LattePanda. De plus, une modification du répertoire par défaut du serveur Apache permet de rediriger l'utilisateur vers la page que l'on souhaite, ou alors provoquer une erreur.

V. Styles Alternatifs

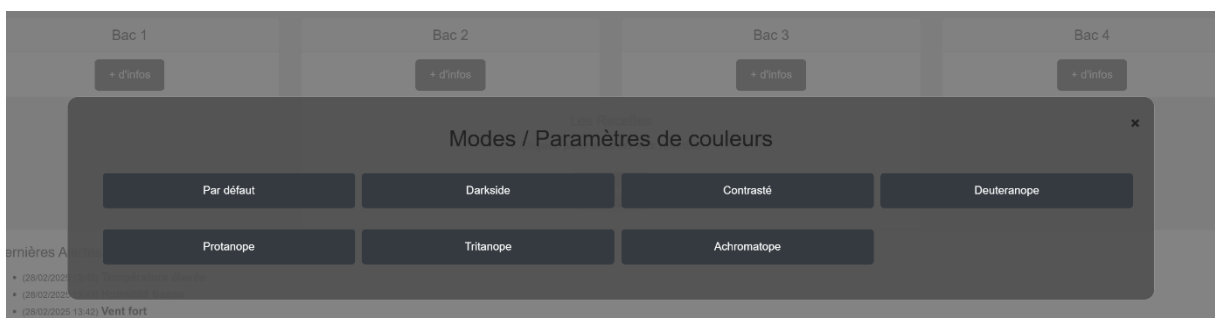
Bien que l'interface Web reste plutôt sobre et n'utilise que très peu de couleurs proches, il est tout de même important de s'intéresser au confort d'utilisation du site pour chaque utilisateur. Cela inclut donc aussi les personnes atteintes de troubles de la vue comme du daltonisme, ou simplement une vue réduite demandant plus d'effort pour lire du texte coloré sur un fond clair. C'est pourquoi plusieurs modes couleurs ont alors été intégrés à l'interface Web.

Ces modes de couleurs sont accessibles depuis un bouton « Mode » situé dans le coin supérieur droit de l'interface.



Bouton « Mode » dans le coin supérieur droit

Ce bouton ouvre alors le menu suivant :



Menu de sélection du mode

L'appui sur l'un de ces boutons émet une requête SQL qui modifie la rubrique mode et recharge la page avec les bonnes couleurs.

1. Daltonismes

Les premiers modes de vue qui ont été ajoutés sont ceux qui correspondent aux 4 grands types de trouble de la vision des couleurs. À savoir :

- Deutéranopie : Absence de vert dans les mélanges de couleurs
- Protanopie : Absence de rouge dans les mélanges de couleurs
- Tritanopie : Absence de bleu dans les mélanges de couleurs
- Achromatopsie : Absence totale de couleurs dans les mélanges, vision en nuances de gris

Les cas des anomalies (Deutéranomalie, Protanomalie et Tritanomalie) n'ont pas eu de modes spécifiques car il s'agit de variation des trois catégories précédentes avec des dysfonctionnements moins sévères.

Il y a donc un mode de couleur pour chacun des cas listés. Ces modes sont définis afin de supprimer les couleurs qui sont absentes, cependant, dans notre cas, seule la couleur bleue pose réellement un problème. Les modes Deutéranope et Protanope se sont alors vus attribuer des couleurs sensées être plus facilement visibles par les personnes qui en sont atteintes.

Pour ce qui est du mode achromatope, il s'agit alors d'un mode sans couleurs et qui met en place des teintes de gris, de blanc et de noir afin de faciliter la lecture.

2. Contraste élevé

Un autre mode important est celui présentant un contraste élevé. En effet, ce mode cible les personnes qui rencontrent des difficultés à lire sur un écran.

Dans ce mode, les textes sont alors en blanc ou en jaune et placés sur un fond noir ou orange. Cela permet d'attirer l'œil vers les informations importantes. Il n'y a pas de jeux d'ombres ou de relief comme sur les autres modes afin de concentrer l'attention du lecteur sur les éléments textuels.

3. Mode nuit ou sombre

Le dernier mode alternatif disponible est alors le mode sombre. L'objectif de ce dernier est de proposer une expérience plus reposante pour l'œil. En effet, l'utilisation de l'interface dans une pièce sombre, ou pendant la nuit, peut provoquer un certain inconfort visuel et accélérer la fatigue en raison de la présence d'un fond clair et de plusieurs couleurs vives.

Le mode sombre met donc en place un fond noir et des couleurs froides pour permettre à l'œil de se reposer lors de la lecture dans un contexte de basse luminosité.

Ce mode ne répond donc pas à une demande d'accessibilité, mais sa présence permet d'augmenter le confort des utilisateurs réguliers.

VI. Normes RGG/RGAA

Le projet de SAÉ a été séparé en plusieurs parties. La partie qui va être étudiée ici est la partie RGG/RGAA (Référentiel général d'amélioration de l'accessibilité).

Il faut donc savoir à quoi correspond cette partie du projet pour pouvoir le mettre en place. L'objectif du projet de SAÉ est de réaliser un site internet permettant de visualiser l'évolution des plantes. Pour chaque site internet, il est obligatoire de permettre l'accès aux personnes en situation de handicap pour tous les services publics en France. C'est ce qu'on appelle en France les normes RGG/RGAA. Ces normes, RGAA (Référentiel Général d'Amélioration de l'Accessibilité) et RGG (Référentiel Général de Gouvernance) sont des règles utilisées en France pour garantir l'accessibilité et la gouvernance des services publics numériques. Plus précisément, le RGAA est un référentiel technique qui définit les critères et les tests à appliquer pour vérifier l'accessibilité des services publics numériques, tandis que le RGG est un cadre de référence qui vise à améliorer la gouvernance des systèmes d'information des administrations publiques. Ces deux principales normes ont des objectifs différents mais sont complémentaires. L'objectif principal du RGAA est d'assurer l'accessibilité à tous, comme les malvoyants, les malentendants, etc..., alors que l'objectif du RGG est d'assurer une gestion efficace des systèmes d'information publics. Il faut savoir que le RGAA définit trois niveaux de conformité (le niveau A, AA et AAA), sachant que le niveau "AA" est le niveau généralement requis. Ces normes (RGG/RGAA) sont présentes à différentes échelles (française, européenne, internationale).

Initialement, pour notre projet, il fallait se baser sur les normes françaises (vu que notre site sera hébergé en France sur le Lattepanda) et sera utilisé par l'IUT (GEII). Mais en recherchant des sites pour pouvoir vérifier si le site respecte toutes les normes et n'en trouvant, il a été décidé de respecter les normes RGG/RGAA européennes, car plusieurs sites existent permettant de tester notre site comme (<https://www.w3.org/WAI/test-evaluate/tools/list/>, ou encore <https://wave.webaim.org>).

À partir de ce moment-là, le travail sur cette partie s'est divisé en 3. Une personne a chacun eu sa tâche attribuée. L'un d'entre nous s'est occupé de créer différents modes sur la page web pour les personnes atteintes de daltonisme. Plusieurs de ces daltonismes sont présents sur notre site web (les daltonismes les plus fréquents). Une autre personne a réfléchi au design du site web avec quelques animations, logos, l'organisation des éléments présents sur la page web.

En même temps que la réalisation des deux tâches précédentes, une autre tâche a été réalisée. Cette tâche consistait à récupérer le document réunissant les normes RGG/RGAA européennes :

(https://accessibilite.numerique.gouv.fr/doc/fr_301549v020102p.pdf)

Et de le lire tout en sélectionnant des normes. Le minimum de normes demandé par le cahier des charges était de 10 normes. Dans ce document, un grand nombre de règles sont énumérées pour l'accessibilité aux personnes en situation de handicap, que ce soit pour un site web, ou bien même pour entrer dans un bâtiment.

Dans le cadre de notre projet, seules les parties 9 et 11, qui représentent la partie Web et la partie Logiciel, nous concernent. Il a donc fallu lire toutes les normes présentes dans ces deux parties, les assimiler et réfléchir à comment les inclure dans le site internet. Après avoir listé les différentes normes jugées les plus importantes dans un document PDF fourni en annexe de ce compte rendu, il est possible de retrouver 42 normes sélectionnées.

Afin de pouvoir vérifier si les normes ont été mises en place, un autre document, Excel cette fois-ci, aussi fourni en annexe, permet de connaître le contexte de la norme (avec un mot clé correspondant à cette même norme), le numéro de la norme correspondant au PDF ainsi que son état sur le site web (si elle est mise en place ou non).

Thèmes de la norme	Numéro de la norme	Mis en place	à mettre en place	Ne sera pas mis en place	Réflexion de la mise en place de la norme
texte	1		x		
texte	2	x			
texte	3	x			
texte	4				x
texte	5		x		
texte	6	x			
texte	7	x			
texte	8			x	
texte	9			x	
texte/erreur	10	x			
non-textuel	11	x			
non-textuel	12	x			
non-textuel	13	x			
non-textuel	14		x		
navigation	15	x			
navigation	16	x			
navigation	17	x			
visu	18	x			
visu	19			x	
visu	20	x			
visu	21			x	
visu/couleur	22			x	
changement contexte	23	x			
changement contexte	24	x			
tech assistance	25	x			
tech assistance	26	x			
tech assistance	27		x		
raccourci clavier	28		x		
raccourci clavier	29		x		
clavier	30		x		
focus clavier	31		x		
clignotement	32		x		

Exemple du document Excel présentant l'avancée des normes présentes sur le site web

Actuellement, le projet n'est pas totalement terminé et un grand nombre de normes n'ont pas été vérifiées ou bien mises en place (il faut donc se référer au tableau Excel). Il faudra donc, pour les futurs étudiants travaillant sur le projet, mettre en place les normes sélectionnées manquantes.

VII. Conclusion

Ce projet aura permis de mettre en œuvre les capacités du groupe à travailler dans une équipe de grande taille et donc de tester la capacité de chacun à effectuer le travail demandé, et ce dans les temps. Cela a aussi permis de comprendre le travail de gestion d'équipe et de coordination des moyens qui est effectué dans ce genre de phase.

D'un point de vue technique, le travail demandé a été mené à bien par les deux équipes qui ont travaillé conjointement jusqu'au bout, résultant en un cahier des charges respecté et un travail fini dans les temps.