Le distributeur d'EPI

COME BONNEUIL
RAPHAEL CHOMAT
THIBAUD MARGOTTEAU
ETHAN OZBEY

Table des Matières

Table des Matières 2
1. Présentation 3
A. Contexte 3
B. Besoin 3
C. Solution/Produit 3
D. Développement Durable 4
2. Cahier des charges 4
A. Diagramme de contexte 4
B. Diagramme fonction principale/ contrainte 5
C. Diagramme d'exigence 6
D. Diagramme d'utilisation 8
3. Conception préliminaire 9
A. Recherche de composants 9
a. Composants électroniques et informatiques 9
Microcontrôleur 9
Interface utilisateur 9
Identification et contrôle d'accès 9
Base de données et serveurs 9
b. Composants mécaniques et moteurs 9
Actionneurs 9
c. Logiciels et développement 9
d. Autres éléments 9
B. Schéma synoptique 10
C. Diagramme Sysml BDD 11
D. Diagramme Sysml IBD 12
F. Chaine d'Information et de Puissance 13
Chaîne d'information 13
Chaîne de puissance 13
4. Conception détaillée 14
A. Composants électroniques et informatiques 14

B. Base de données et serveur 18
Réalisation et test :..... 24
C. IHM..... 26
Pour quoi une IHM : 26
Maquette : 26
Les différents pages pour le code : 27
Explication d'une partie de code : 28
Conclusion : 33

1. Présentation

A. Contexte

Les équipements de protection individuelle (EPI) sont essentiels pour garantir la sécurité des salariés, en particulier dans les secteurs à risque de construction, l'industrie ou la manutention. La gestion des EPI présente donc un défi important à résoudre pour de nombreuses entreprises, sur le plan financier et en gestion des ressources.

Chaque salarié, en fonction de son poste et de ses responsabilités, aura des besoins spécifiques en matière d'EPI. Dans de nombreuses entreprises, la distribution de ces équipements est encore effectuée de manière manuelle, souvent sans prise en compte du statut ou du grade des employés. Cela peut donc entraîner une mauvaise gestion des stocks, une distribution inappropriée (par exemple, des EPI de mauvaise taille ou non adaptés aux besoins spécifiques de certains postes), ainsi que des pertes importantes.

B. Besoin

Les entreprises doivent gérer les EPI de manière à la fois écologique et économique, en tenant compte du statut des employés. Il est également crucial de faciliter l'accès aux EPI pour chaque salarié en fonction de ses besoins.

Problématique : facilité l'accès aux équipements de sécurité tout en minimisant l'impact environnementale ?

C. Solution/Produit

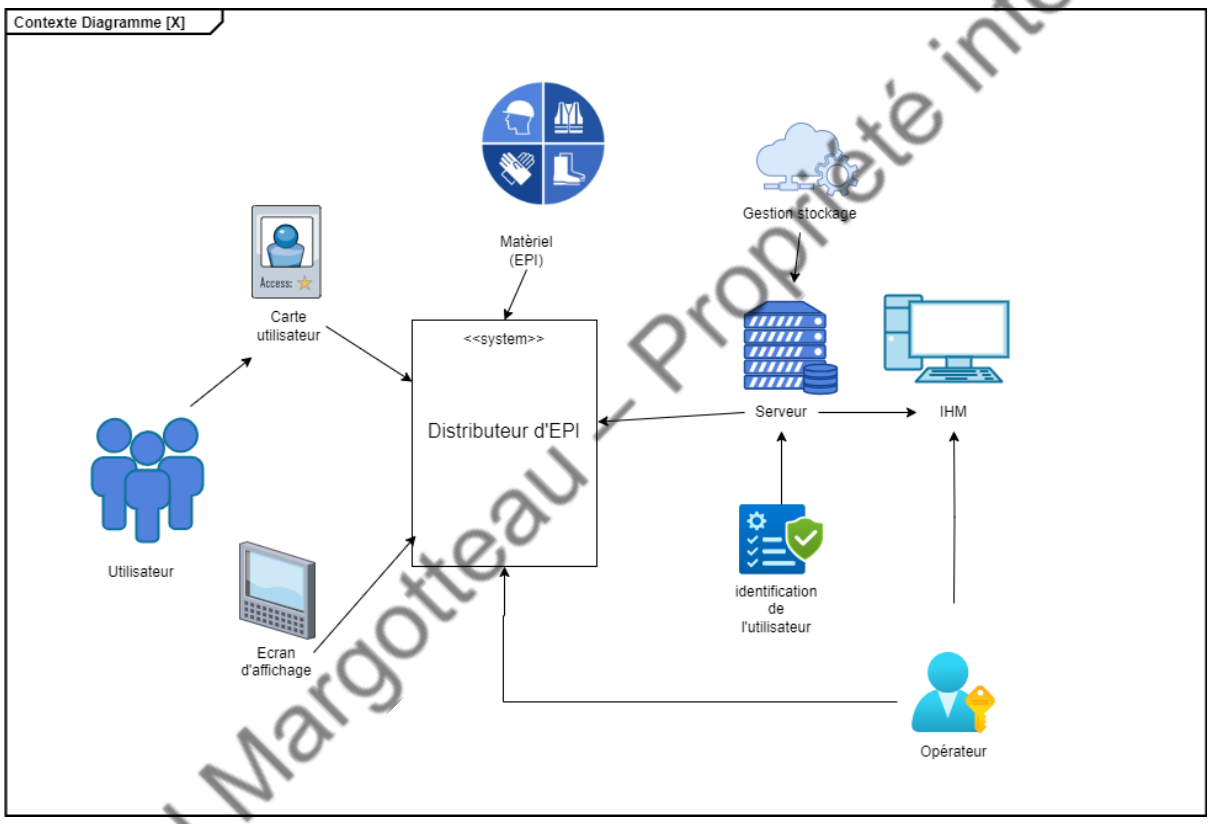
Pour répondre aux problématiques de gestion et de distribution des EPI, nous proposons un distributeur automatique d'EPI intelligent et sécurisé. Ce système innovant permet de fournir des équipements de protection individuelle de manière automatique et personnalisée, en fonction du statut et du grade de chaque salarié.

D. Développement Durable

Ce système aide les entreprises à réduire les pertes liées à la distribution des EPI, tout en contribuant à une gestion plus durable des ressources. Il permet également d'améliorer la sécurité des employés en garantissant un accès rapide et ciblé aux équipements de protection.

2. Cahier des charges

A. Diagramme de contexte



Ce diagramme de contexte illustre les interactions entre le distributeur d'Équipements de Protection Individuelle (EPI) et les différents utilisateurs du système.

L'utilisateur accède au distributeur en utilisant une carte dédiée qui permet son identification. Un écran d'affichage fournit des informations en temps réel sur son profil et le matériel disponible. Une fois authentifié, l'utilisateur peut retirer les EPI nécessaires.

L'opérateur a un accès privilégié au système via l'IHM, lui permettant d'administrer les stocks et d'effectuer des ajustements si nécessaire. Il peut modifier les paramètres du distributeur, gérer les autorisations d'accès et assurer le bon fonctionnement du

dispositif. Comment facilité l'accès aux équipements tout en minimisant l'impact environnementale.

B. Diagramme fonction principale/ contrainte

	Nom	Critères	Niveaux	Flexibilité
FP1	Distribuer des EPI	Sélectionner la taille et les Epi	Digicode	F0
FP2	Identifier	Scanner la carte de l'utilisateur	Statue de la personne	F0
FP3	Stocker les EPI	Permettre un stockage d'EPI	500EPI	F0
FC1	Afficher Information distributeur	Permettre à l'utilisateur de voir son profil (nom, prénom, grade, crédit)	Directeur à Stagiaire	F2
		Permettre à l'utilisateur de voir le nombre d'équipement qu'il peut retirer.	0 à 10	
FC2	Stocker les donnés	Gérer les stocks	Lunettes, gants, masque, boule quies etc.	F0
		Gérer les tailles	S à XXXL	
		Gérer les employés (ajouter, supprimer, statuts, etc.)	10 à 100 Utilisateur	
FC3	Faciliter l'accessibilité	Interface Web en local	Ecran OLED	F1
			Modifier les données	

Ce tableau présente les fonctions principales et contraintes associées au fonctionnement d'un distributeur d'Équipements de Protection Individuelle (EPI). Il définit les différentes fonctionnalités du système, leurs critères et leurs niveau exigées.

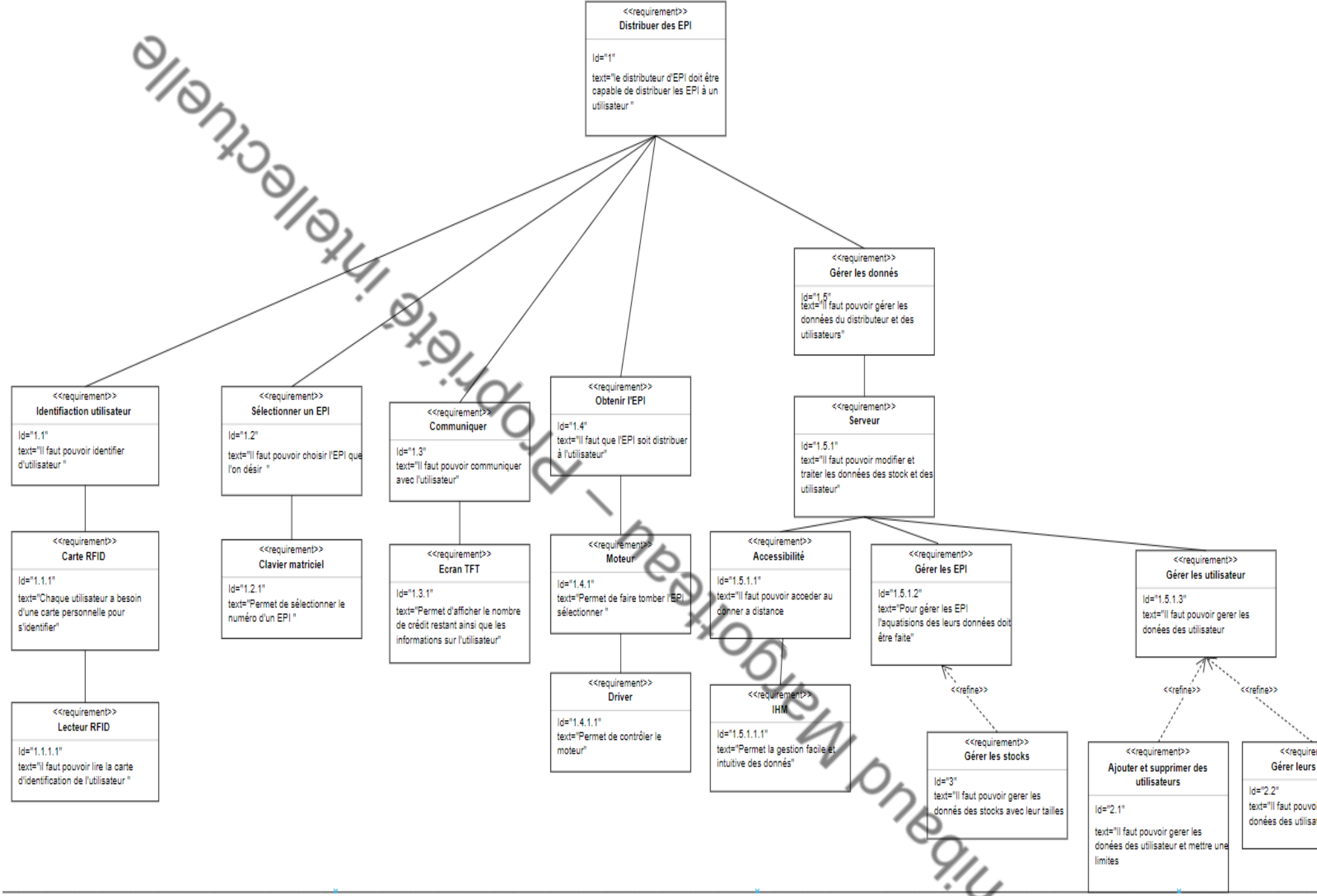
Le distributeur doit permettre aux utilisateurs de sélectionner la taille et le type d'EPI à retirer, avec un accès sécurisé par digicode. L'identification repose sur la lecture de la carte de l'utilisateur. Le stockage des EPI est limité à une capacité de 500 unités.

Le distributeur affiche des informations essentielles aux utilisateurs, notamment leur identité (nom, prénom, grade) ainsi que le nombre d'équipements qu'ils peuvent encore retirer. (Crédits restants)

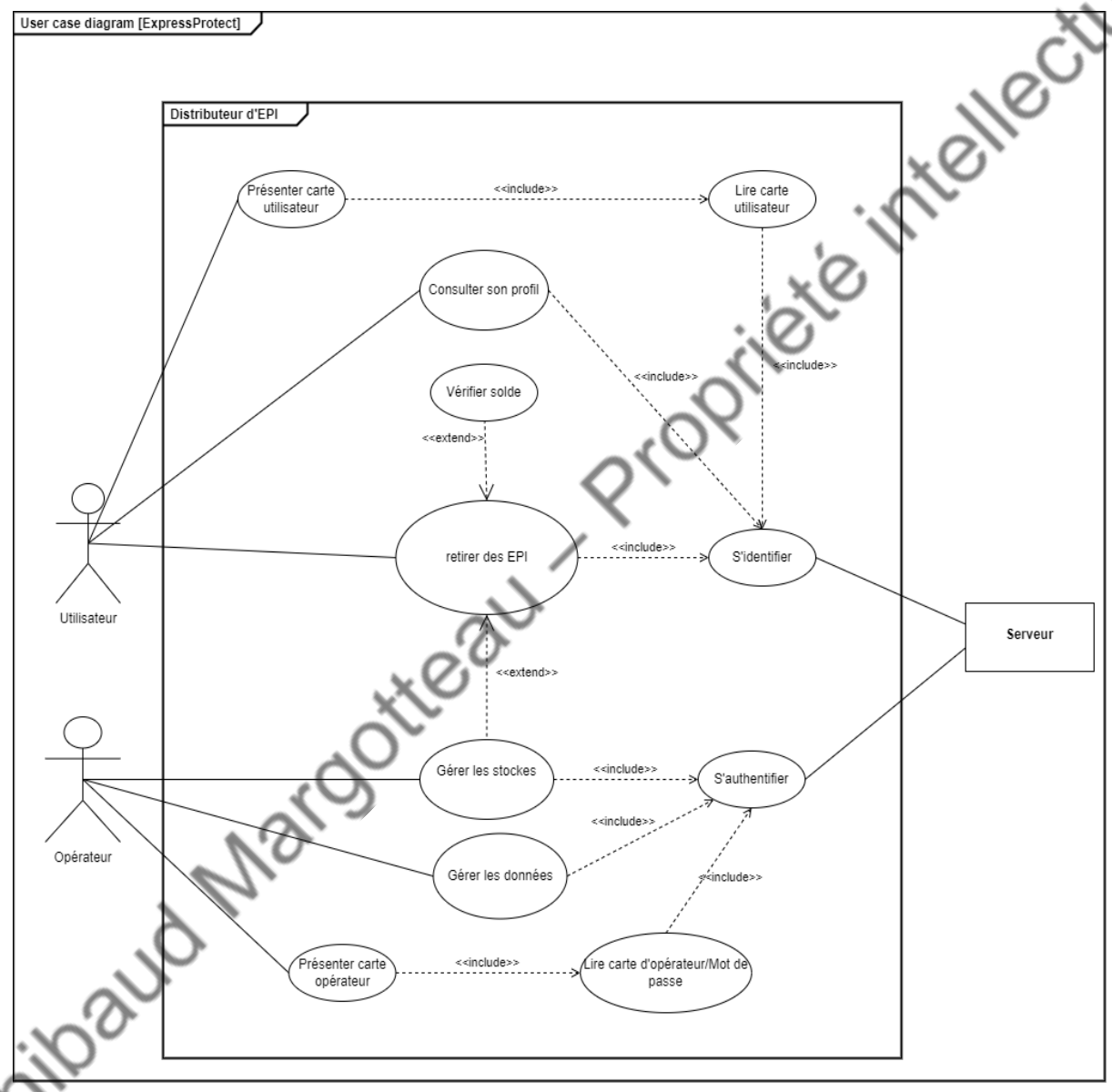
La gestion des données concerne le suivi des stocks d'EPI disponibles, avec une classification par type (lunettes, gants, masques, etc.) et par taille (de S à XXXL). Le système doit également gérer les employés en permettant l'ajout, la suppression et la mise à jour de leurs statuts, dans une plage allant de 10 à 100 utilisateurs.

L'accessibilité est améliorée pour les utilisateurs par un écran OLED, et une interface web locale (IHM) pour l'opérateur, permettant de modifier les données.

C. Diagramme d'exigence



D. Diagramme d’utilisation



Ce diagramme d'utilisation montre les différentes actions possibles ainsi que les relations entre ces actions.

L’utilisateur peut interagir avec le distributeur en présentant sa carte, ce qui déclenche la lecture des informations par le système. Il peut ensuite consulter son profil pour vérifier son solde avant de retirer des EPI. L’action de retrait implique une identification qui est prise en charge par le serveur.

L’opérateur joue un rôle de gestionnaire des stocks et des droits relatifs à chaque utilisateur. Il s’authentifie un mot de passe depuis une IHM, ce qui lui permet d’accéder aux fonctionnalités de gestion des stocks et des données.

3. Conception préliminaire

A. Recherche de composants

a. Composants électroniques et informatiques

Microcontrôleur

- **ESP8266** : Carte de contrôle pour gérer les composants du projet, et pour se connecter à internet.

Interface utilisateur

- **Clavier matriciel 4x4** : Saisie de chiffres et lettres.
- **Écran LCD :** Affichage des informations à l'utilisateur.

Identification et contrôle d'accès

- **Lecteur RFID RC522** : Lecture et écriture sur cartes/badges RFID.
- **Carte RFID** : Stockage et transmission des données d'identification.

Base de données et serveurs

- **MySQL** : Gestion des données.
- **Always Data :** Plateforme de développement web pour l’accès et gestion des données.

b. Composants mécaniques et moteurs

Actionneurs

- **Moteur pas à pas Nema 17** : Permet le mouvement d’un système mécanique.
- **Module L298N** : Contrôle du moteur pas à pas.

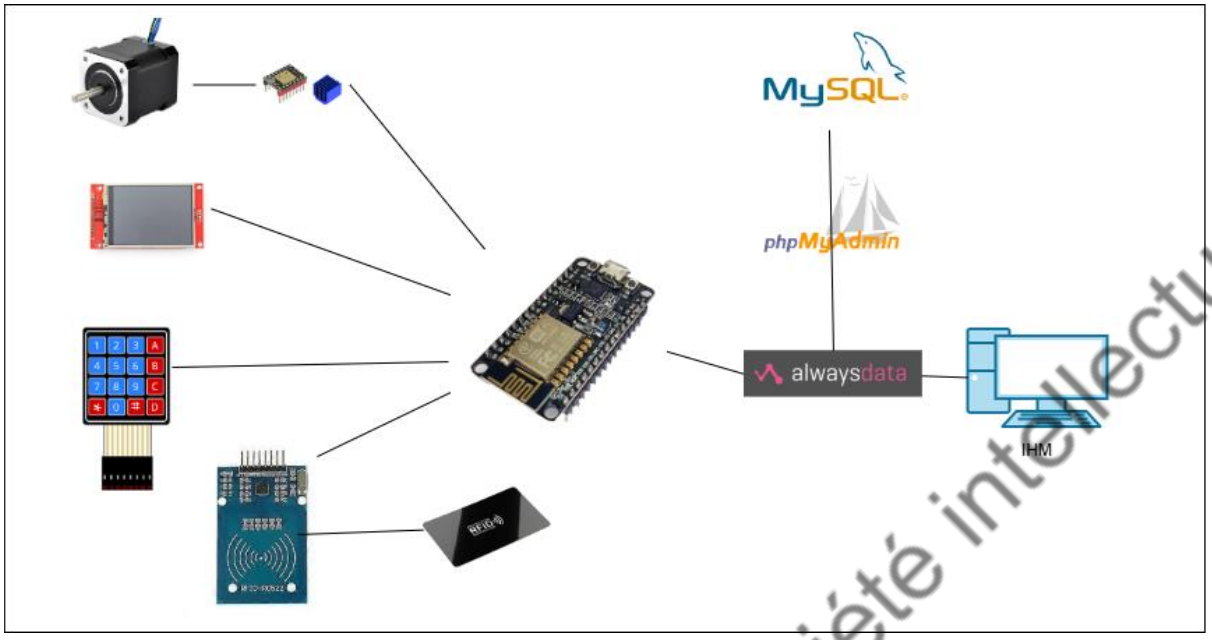
c. Logiciels et développement

- **HTML/CSS** : Structure et style de l’interface web.
- **PHP/SQL** : Gestion des données et communication avec la base de données.

d. Autres éléments

- **Divers** : Regroupe les éléments annexes nécessaires à l’assemblage.

B. Schéma synoptique

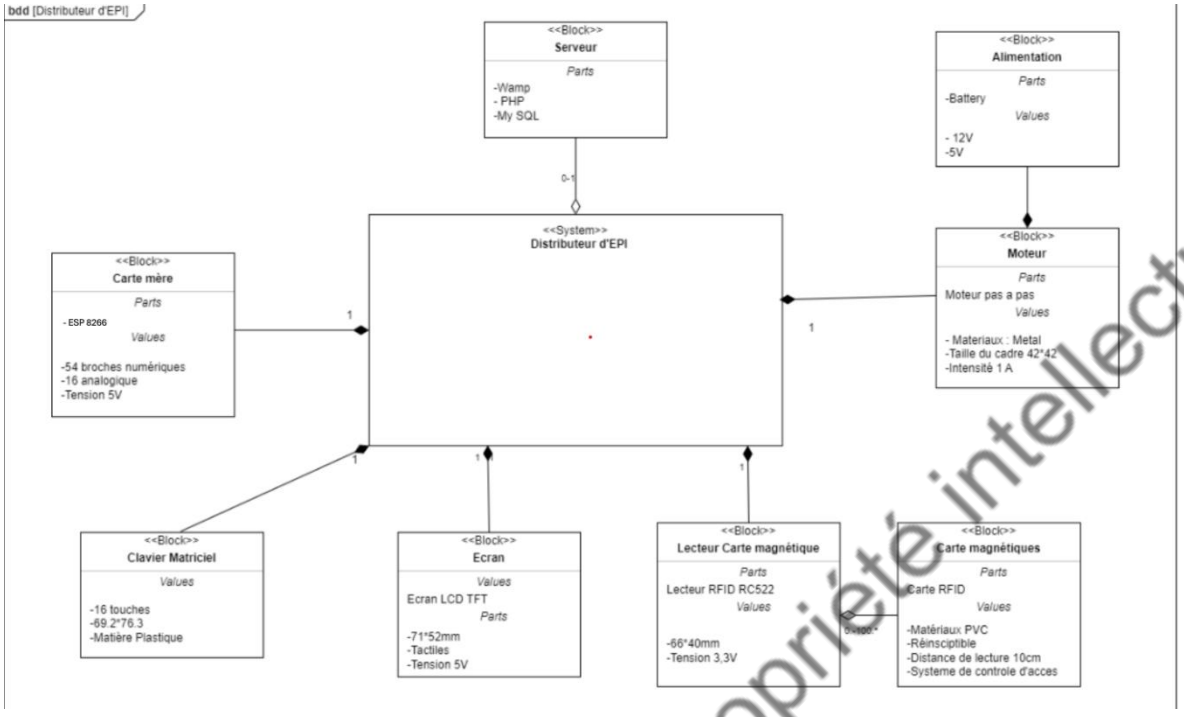


Ce schéma synoptique représente l'architecture matérielle et logicielle du distributeur d'Équipements de Protection Individuelle (EPI). Il met en évidence les composants électroniques, les systèmes de gestion de données et l'interface utilisateur.

Au cœur du système se trouve une carte Arduino, qui centralise les connexions et assure le contrôle des différentes interactions. Elle est reliée à plusieurs périphériques essentiels : un moteur pas à pas permettant la distribution des EPI, un écran d'affichage pour informer l'utilisateur, un clavier matriciel pour la saisie des informations, ainsi qu'un module RFID chargé de l'identification des utilisateurs via des cartes ou badges.

Un serveur, utilisant WampServer, héberge une base de données MySQL administrée via phpMyAdmin. Il assure le stockage et la gestion des informations, notamment les identifiants des utilisateurs et les niveaux de stock des EPI. Les données sont accessibles et manipulables via une interface homme-machine (IHM) sur un ordinateur.

C. Diagramme Sysml BDD



Ce diagramme SysML BDD (Block Definition Diagram) décrit la structure du distributeur d'Équipements de Protection Individuelle (EPI) en détaillant ses composants matériels et leur organisation. Il permet d'identifier les éléments constitutifs du système ainsi que leurs caractéristiques techniques.

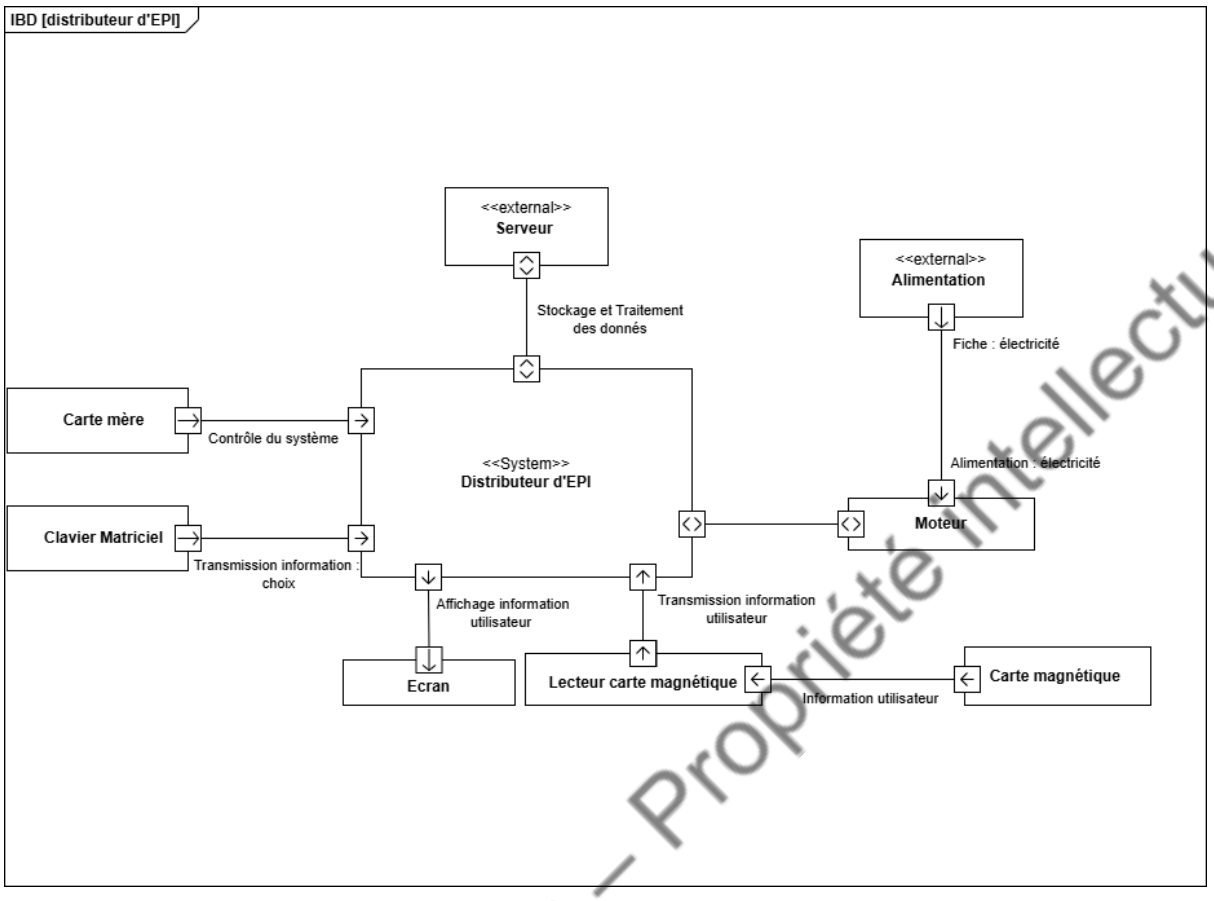
Le cœur du système est le distributeur d'EPI, qui interagit avec plusieurs sous-systèmes essentiels. Il est connecté à un serveur MySQL, qui assure la gestion des données et le stockage des informations relatives aux utilisateurs et aux équipements disponibles.

L'alimentation du système est assurée par un bloc d'alimentation fournissant une tension de 12V qui permet d'alimenter le moteur pas à pas. Ce moteur fonctionne avec 12V et un courant de 1A. Son activation permet de distribuer l'EPI a l'utilisateur.

Le contrôle du système repose sur une carte mère Arduino Méga, dotée de 54 sorties numériques et fonctionnant sous 5V.

L'interface utilisateur est composée de plusieurs modules. Un clavier matriciel composé de 16 touches qui permet de sélectionner l'équipement souhaité ainsi qu'un écran LCD qui affiche les informations de l'utilisateur. Elle inclue également un lecteur de cartes magnétiques RFID qui est alimenté en 3.3V. Il est utilisé pour l'identification des utilisateurs via des cartes RFID personnelle à chacun.

D. Diagramme Sysml IBD



Le distributeur d'EPI est au centre du schéma et interagit avec plusieurs éléments essentiels. Il est alimenté électriquement par un bloc d'alimentation, qui fournit l'énergie nécessaire à son fonctionnement, notamment au moteur chargé de la distribution des équipements.

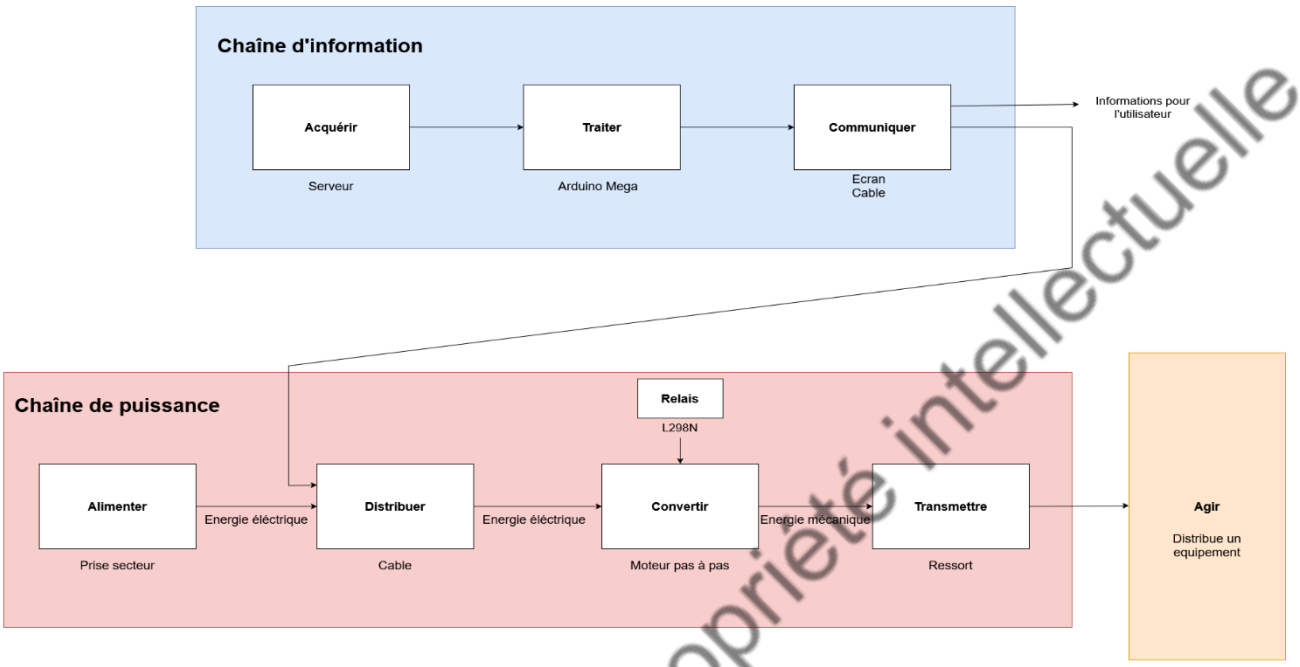
Le serveur est responsable du stockage et du traitement des données. Il gère les informations des utilisateurs et le statut des stocks, permettant une gestion optimisée du système.

La carte mère assure le contrôle global du distributeur, en supervisant les interactions entre les différents composants.

Le clavier matriciel, reçoit les instructions de l'utilisateur (l'EPI qu'il a choisie) et les transmet pour que l'EPI voulu soit sélectionner.

L'écran, qui affiche les informations pertinentes, et reliée à un lecteur de carte magnétique, qui permet l'identification sécurisée des utilisateurs via des cartes magnétiques. Ce lecteur transmet les informations d'authentification à la carte mère, qui vérifie l'accès et autorise ou non le retrait des EPI.

F. Chaine d'Information et de Puissance



Ce diagramme représente la chaîne d'information et de puissance du distributeur d'Équipements de Protection Individuelle (EPI). Il est divisé en deux sections principales : la chaîne d'information et la chaîne de puissance.

Chaîne d'information

La chaîne d'information, en bleu, gère le traitement des données nécessaires au fonctionnement du distributeur :

1. **Acquérir** : Le serveur gère les informations, notamment les données des utilisateurs et des stocks.
2. **Traiter** : L'Arduino Mega vérifie les informations d'identification, vérifie les crédits de l'utilisateur et active le moteur.
3. **Communiquer** : Les informations de l'utilisateur s'affiche sur l'écran et permet également d'afficher les messages pertinents, comme le solde d'équipements disponibles ou les instructions d'utilisation.

Chaîne de puissance

La chaîne de puissance, en rouge, assure l'alimentation et la distribution physique des EPI :

1. **Alimenter** : L'énergie est fournie par une prise secteur, qui alimente le système en électricité.

2. **Distribuer** : Cette énergie est transmise via des câbles aux composants nécessaires.
3. **Relais** : Il agit d'un driver qui permet de piloter le moteur pas à pas.

4. **Convertir** : Le moteur pas à pas transforme l'énergie électrique en énergie mécanique.
5. **Transmettre** : Le ressort permet d'amplifier cette énergie mécanique pour faire tomber l'équipement dans la trappe du distributeur.
6. **Agir** : Le système distribue physiquement un équipement à l'utilisateur.

4. Conception détaillée

A. Composants électroniques et informatiques

Microcontrôleur : L'esp 8266 est utilisé pour connecter le système (le distributeur) a internet grâce à une connexion wifi, car il est nécessaire d'avoir une connexion à internet pour pouvoir communiquer avec la base de données.

```
//on importe la bibliotheque pour le wifi
#include "WiFi.h"
int bouton = 5;
int press;
//variables pour le nom de reseau et le mot de passe
const char* reseau     = "Saint-Nicolas Peda";
const char* motDePasse =

void setup() {
  // on initiliaise le wifi en tant que station
  WiFi.mode(WIFI_STA);
  //si une connexion était deja en cours, on se deconnecte
  WiFi.disconnect();
  //initilisaiton de la liaison série
  Serial.begin(9600);

  //Initialisation du bouton qui déclenchenra la connexion au wifi
  pinMode(bouton, OUTPUT);
  pinMode(5, INPUT_PULLUP);
}

void loop(){

//On met la valeur(0 si appuyé,sinon 1)
press = digitalRead(bouton);

//On lance la connexion au Wifi si le bouton est appuyé
if (press == 0) {
  ConnexionWifi(reseau, motDePasse).
}
}
```

Voici donc ci-dessus le programme permettant la connexion de l'ESP 8266 à internet grâce à une connexion wifi.

En effet dans ce programme, quand un bouton déclaré on préalable va être déclenché la connexion (ici au wifi du Lycée Saint-Nicolas) démarrera

Identification et contrôle d'accès : l'identification et le contrôle d'accès d'un travailleurs se fait grâce à un système RFID (Radio Fréquency identification), composé d'une carte et d'un lecteur de carte qui sera relié au microcontrôleur.

```
#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN    16
#define SS_PIN     4

MFRC522 mfrc522(SS_PIN, RST_PIN);
void setup() {
  Serial.begin(9600);
  mfrc522.PCD_Init();   // Init MFRC522
  mfrc522.PCD_DumpVersionToSerial();  // Show details of PCD - MFRC522 Card Reader details
}

void loop() {
  // Reset the loop if no new card present on the sensor/reader. This saves the entire process when idle.
  if ( ! mfrc522.PICC_IsNewCardPresent()) {
    return;
   Serial.print("carte presenté");
  }

  // Select one of the cards
  if ( ! mfrc522.PICC_ReadCardSerial()) {
    return;
  }

  // Dump debug info about the card; PICC_HaltA() is automatically called
  mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
}
```

Dans ce programme : après avoir configuré les bibliothèques nécessaires et les pin le programme fait en sorte que quand on passe un badge RFID devant le lecteur de carte RFID l'identifiant de la carte s'affiche. On peut donc après grâce à l'identifiant de la carte la relié à un utilisateur et ses données de notre base de données.

Clavier matricielle : Dans un distributeur un clavier matriciel est utilisé pour permettre à l'utilisateur de choisir l'EPI qu'il souhaite avoir.

```
#include "Adafruit_Keypad.h"
#define KEYPAD_PID3844

#define R1    11
#define R2    8
#define R3    9
#define R4    10
#define C1    12
#define C2    14
#define C3    6
#define C4    7

#include "keypad_config.h"

Adafruit_Keypad customKeypad = Adafruit_Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS);

void setup() {
  Serial.begin(9600);
  customKeypad.begin();
}
void loop() {
  // put your main code here, to run repeatedly:
  customKeypad.tick();

  while(customKeypad.available()){
    keypadEvent e = customKeypad.read();
    Serial.print((char)e.bit.KEY);
    if(e.bit.EVENT == KEY_JUST_PRESSED) Serial.println(" pressed");
    else if(e.bit.EVENT == KEY_JUST_RELEASED) Serial.println(" released");
  }
}
```

Un clavier matriciel a 8 sorties, chacune correspondant a une ligne ou une colonne. On commence donc par définir le pin relié à chaque ligne ou colonne, pour ensuite, grâce à « Adafruit_Keypad customKeypad = Adafruit_Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS) » les transformer en tableau qui sera notre clavier. Ensuite le programme s'occupe d'afficher quand une des touches est pressé, la touche correspondante.

Ecran LCD I2C : Dans notre distributeur l'écran va permettre d'afficher les informations de l'utilisateur (après qu'il est scanné sa carte) comme notamment : nom, prénom, poste dans l'entreprise et crédit restant, ou relative au distributeur, comme le succès de la demande d'une EPI ou du refus.

On utilise un écran LCD (Liquid Crystal Display) I2C (Inter-Integrated Circuit) qui nous permettra de facilement afficher ces informations, tout en ayant une consommation faible grâce au LCD et une faible utilisation de pin grâce à l'I2C.

Moteur pas à pas Nema17 : Le moteur pas à pas Nema17 permet un contrôle précis de la rotation grâce à ses pas fixes (1,8°), sans capteur de position. Il offre un couple

suffisant (~40 N.cm) pour faire tourner un ressort et maintenir sa position. Sa vitesse est ajustable, ce qui évite les à-coups lors de la distribution. Sa compacité (42x42 mm) facilite son intégration dans un boîtier réduit.

Pont en H - L298N : Le L298N est un pont en H qui sert à contrôler un moteur (sens et vitesse) depuis une carte comme un Arduino ou un ESP8266. Il permet de faire tourner le moteur dans un sens ou dans l'autre, utile si le ressort doit revenir en arrière. Il supporte assez de tension et de courant pour un moteur comme le Nema17. Il est facile à câbler et peut aussi alimenter le microcontrôleur en 5V. C'est un composant simple et pratique pour notre système. Il utilise 4 transistors : selon lesquels sont activés, le courant passe dans un sens ou dans l'autre. Donc, le moteur peut tourner à gauche ou à droite.

L'objectif final est de fusionner ce code moteur avec la partie RFID : lorsque l'identification par badge est validée via le serveur, le moteur effectue automatiquement une rotation complète pour libérer une EPI. Cela permettra d'automatiser complètement la distribution, tout en garantissant que seuls les utilisateurs autorisés puissent retirer un équipement.

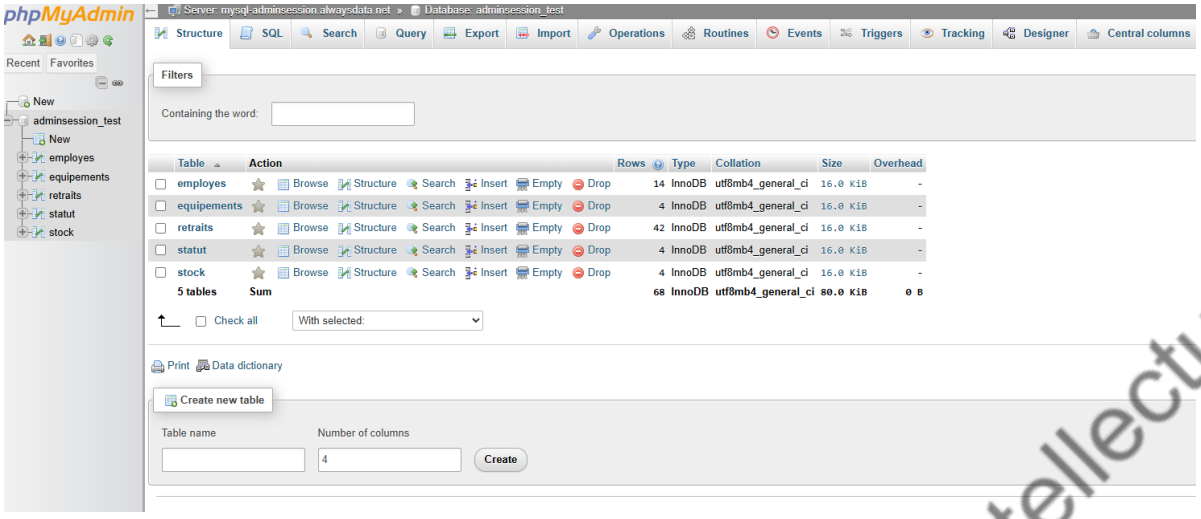
B. Base de données et serveur

Pour la partie base de données et serveur, j'ai utilisé :

- PhpMyAdmin : pour créer et gérer la base de données
- FileZilla : pour transférer mes fichiers vers le serveur distant.
- Php : pour communiquer entre l'IHM et la base de donnée

Premièrement, on a créé une base de données qui s'appelle adminsession_test. On a créé différentes tables :

- employes
- equipements
- retraits
- statut
- stock



Voici 3 requêtes SQL pour créer les différentes tables :

Requête SQL pour créer la table employes:

```
CREATE TABLE employes (
  ID INT AUTO_INCREMENT PRIMARY KEY,
  Prénom VARCHAR(30),
  Nom VARCHAR(30),
  statut INT,
  carte_rfid VARCHAR(100)
);
```

Donc en premier, on créer une clé incrémentation unique à chacun, un prénom et nom avec 30 caractères, un statut en INT donc un chiffre qui correspond à l'auto incrémente du statut en question présent dans la table statut.

Requête SQL pour créer la table statut:

```
CREATE TABLE statut (
 id_statut INT AUTO_INCREMENT PRIMARY KEY,
nom_du_statut VARCHAR(50),
max_gants INT,
max_masques INT,
```

max_boules_quieces INT,

max_gilets INT

);

Donc en premier, on créer une clé incrémentation unique à chaque statut, le nom du statut avec 50 caractères, la quantité en chiffre maximal des différents équipements auxquels il a le droit.

Requête SQL pour créer le table retraits :

CREATE TABLE retraits (

id_retrait INT AUTO_INCREMENT PRIMARY KEY,
id_employe INT,

id_equipement INT,

quantite INT,

date_retrait DATETIME DEFAULT CURRENT_TIMESTAMP
);

Donc encore une fois on crée une clé auto-incrémentation par retrait, id_employe qui correspond à l'auto-incrémentation dans la table employe, id_equipement qui correspond à l'auto-incrémentation dans la table equipements, la quantité en chiffre d'EPI pris ainsi que la date de retrait.

Les fonctions de l'IHM :

- Ajouter un employé
- Afficher la liste des employées
- Modifier informations d'un employé déjà enregistrée
- Supprimer un employé
- Ajouter un statut
- Afficher la liste des statuts
- Modifier informations d'un statut déjà enregistrée
- Supprimer un statut
- Afficher les statistiques du nombre de retraits au total et par employées
- Afficher la quantité d'EPI dans le distributeur
- Modifier la quantité d'EPI présent dans le distributeur

Création d'une fonction pour se connecter à la base de données (config.php) :

```php
<?php

$servername = "mysql-adminsession.alwaysdata.net";
$username = "402820";
$password = "123456";
$dbname = "adminsession_test";

// se connecte a la base de données
$conn = new mysqli($servername, $username, $password, $dbname);

// message si la connection a raté
if ($conn->connect_error) {
    die("Échec de la connexion : " . $conn->connect_error);
}
?>
```

Cette fonction sera appelée pour se connecter à la base de données sur page.

Elle permet d'optimiser le code pour réduire sa longueur, facilité les modifications des informations de la base de données. Par exemple, si le mot de passe de la base de données est amené à changer, il suffira de la modifier dans la fonction sur la page config.php et non d'aller sur chaque page pour modifier le mot de passe.

Explication de comment ajouter un employé (ajout_employe.php) :

1. On appelle la fonction pour se connecter à la base de données sachant que les paramètres de connexion sont dans le fichier config.php :

   include 'config.php';

2. On récupère les informations saisies sur la partie formulaire html de la page :

   $prenom = $_POST["prenom"];
   $nom = $_POST["nom"];
   $statut = $_POST["statut"];
   $carte_rfid = $_POST["carte_rfid"];

   POST est une méthode qui empêche le code de s'exécuter à chaque chargement de page.

3. On prépare une requête SQL pour insérer un nouvel employé dans la table employes

   $stmt = $conn->prepare("INSERT INTO employes (Prénom, Nom, statut, carte_rfid) VALUES (?, ?, ?, ?)");

   Les “?” représentent les données que l’on a inséré dans dans le formulaire html.

4. On crée maintenant une liaison avec les données insérées dans le formulaire pour remplacer les “?”

   $stmt->bind_param("ssis", $prenom, $nom, $statut, $carte_rfid);

   Et “ssis” correspond au type de chaque donnés, donc “s” pour string qui correspond à du texte et “i” pour integer qui correspond à un nombre ou chiffre qui est donc l’id auto incrémente ici du statut.

5. On exécute la requête et on affiche un message de validation ou d’erreurs :

```
if ($stmt->execute()) {
   $message = " Employé ajouté avec succès.";
} else {
   $message = " Erreur lors de l'ajout.";
}
```

Pour la page ajouter un statut c’est la même chose sauf que le nom des variables et des tables sont différentes.

Explication de comment supprimer un employé (supprimer_employe.php) :

1. On appelle une nouvelle fois la fonction pour se connecter à la base de données :

   include 'config.php';

2. On récupère l’id de clé auto incrémentée de l'employé concerné par cette suppression :

   $id = $_POST['id'];

3. On prépare une requête SQL qui va supprimer un enregistrement dans la table employes :

   $sql = "DELETE FROM employes WHERE ID = ?";

4. On crée maintenant une liaison avec l'id de l'employé concerné pour remplacer le "?"

   $stmt->bind_param("i", $id);

   Et "i" correspond au type de chaque donnés, donc "i" pour integer qui correspond à un nombre ou chiffre qui est donc l'id auto incrémente ici de l'employé.

5. On exécute la requête :

   $stmt->execute();

Explication de comment afficher la liste des employées (liste_employes.php) :

1. On appelle une nouvelle fois la fonction pour se connecter à la base de données :

   include 'config.php';

2. Récupérer les informations des tables : employés et statut :

   $sql = "SELECT employes.ID, Prénom, Nom, nom_du_statut, carte_rfid
   FROM employes

   JOIN statut ON employes.statut = statut.id_statut";

   La ligne : JOIN statut ON employes.statut = statut.id_statut"; permet d'aller chercher le nom du rôle présent dans la table statut et non pas l'id qui est indiqué dans la table employes. Si on ne fait pas cela, l'affichage serait comme cela :

   Gérard Denis 3 et non Gérard Denis Conducteur

3. On exécute la requête et on récupère les informations dans $result :

   $result = $conn->query($sql);

4. On vérifie les informations reçues :

```
if ($result && $result->num_rows > 0) {
  while ($row = $result->fetch_assoc()) {
    echo "<tr>...</tr>";
  }
} else {
  echo "<tr><td colspan='6'>Aucun employé trouvé.</td></tr>";
}
```

Donc si la requête renvoie des informations, on parcourt chaque ligne avec fetch_assoc(). Fetch_assoc() est une méthode php qui permet de lire les résultats de la requête SQL ligne par ligne et de faire un tableau. Ensuite on génère une ligne différente par employé avec <tr>. Si on ne reçoit aucune information de la base de données alors on affiche : Aucun employé trouvé.

Réalisation et test :

Lors de la mise en place du système, plusieurs difficultés ont été rencontrées en lien avec les composants utilisés, notamment le moteur pas à pas Nema17, le module L298N et le microcontrôleur ESP8266. L'une des principales complications a concerné le branchement des composants entre eux, en particulier entre le L298N et l'ESP8266. En effet, les ressources disponibles en ligne pour ce type de configuration spécifique (Nema17 + L298N + ESP8266) étaient assez limitées et parfois contradictoires. Cela m'a obligé à revérifier de nombreuses fois le câblage, à tester différentes combinaisons de connexions, et à croiser plusieurs schémas pour éviter toute erreur.

Une fois les branchements validés, la phase de test a été facilitée grâce à des exemples de codes disponibles en ligne, qui ont permis de vérifier que le moteur réagissait bien aux signaux envoyés depuis l'ESP8266 via le L298N. Ces exemples m'ont permis d'isoler rapidement les erreurs de câblage ou de configuration dans le code.

L'objectif final est de fusionner ce code moteur avec la partie RFID : lorsque l'identification par badge est validée via le serveur, le moteur effectue automatiquement une rotation complète pour libérer un EPI. Cela permettra d'automatiser complètement la distribution, tout en garantissant que seuls les utilisateurs autorisés puissent retirer un équipement.

Image de la table employée :

	ID	Prénom	Nom	statut	carte_rfid
Edit Copy Delete	1	Kenny	Chomat	2	D7B30519
Edit Copy Delete	7	Ethan	Ozbey	2	E3T22
Edit Copy Delete	11	Rayan	Ben Miled	4	7Y28Y2
Edit Copy Delete	12	Gustave	Lumière	5	XX1995T
Edit Copy Delete	13	Mathurin	Michau	5	X2007PIERRE
Edit Copy Delete	14	Guilherme	Le Quere	5	10OIYUT68YBB
Edit Copy Delete	16	Thibaud	Margotteau	5	E3T22
Edit Copy Delete	18	Oliver	Maillard	2	HNJ57HF
Edit Copy Delete	20	Corentin	Picard	4	DE33HJ75BN
Edit Copy Delete	22	Raphael	CHOMAT	5	00000001

Check all *With selected:* Edit Copy Delete Export

Image de la page ajout_employe.php :

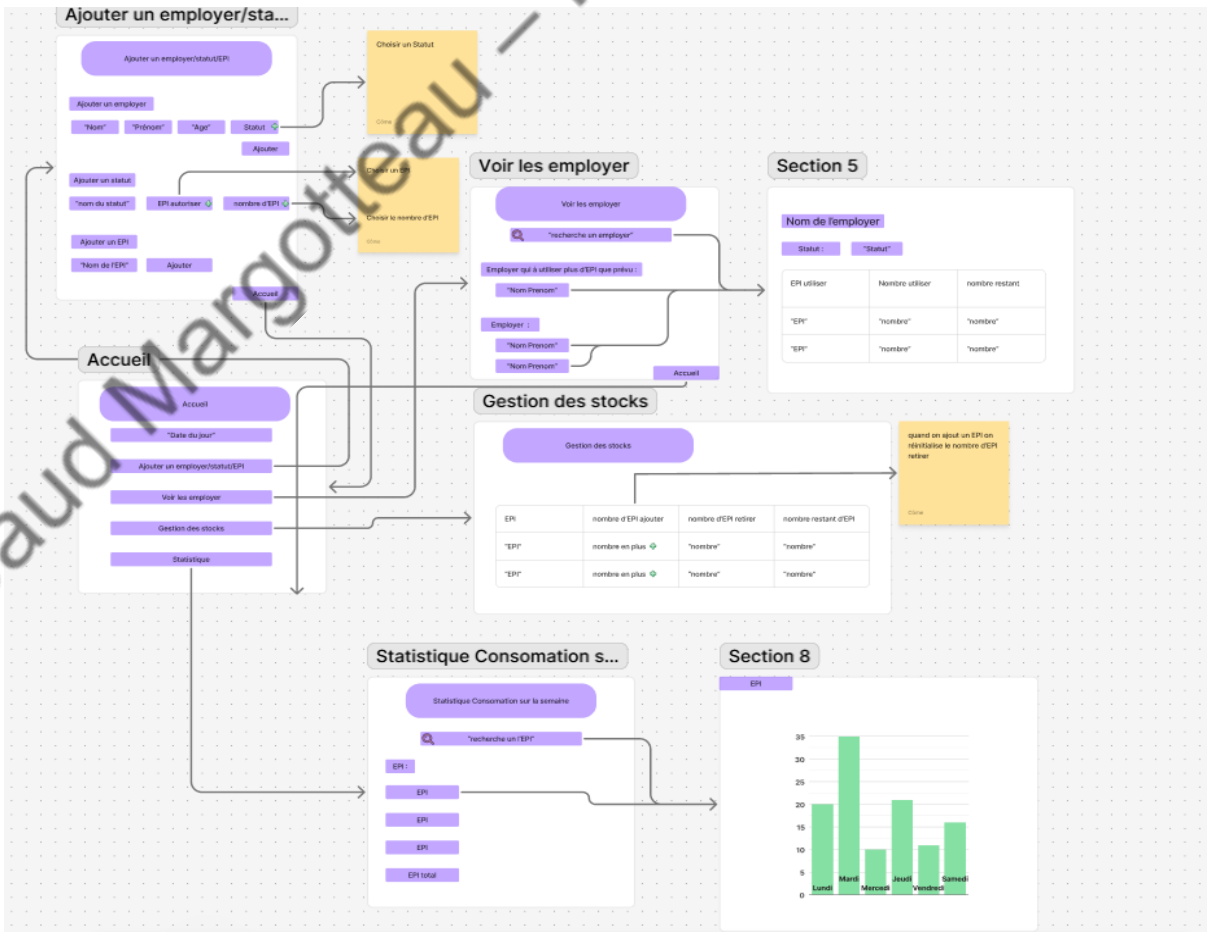
On remarque bien le message de validation de l'enregistrement de l'employé dans la base de données depuis l'IHM.

C. IHM

Pour quoi une IHM :

L'IHM (Interface Homme-Machine) est un élément central de ce projet car elle permet à l'utilisateur d'interagir facilement avec le système sans avoir besoin de compétences techniques. Elle rend l'utilisation du programme plus intuitive et accessible, en simplifiant l'ajout d'employés, la gestion des statuts ou le suivi des équipements de protection. Grâce à des formulaires, des boutons et des messages visuels, l'utilisateur est guidé à chaque étape, ce qui réduit les erreurs et améliore la rapidité d'exécution. L'IHM offre aussi un retour immédiat sur les actions effectuées (comme un message de confirmation), ce qui renforce la compréhension et la confiance dans le système. Elle transforme donc un ensemble de fonctionnalités techniques en une interface claire et compréhensible, indispensable pour une utilisation quotidienne sur le terrain.

Maquette :



Pour réaliser cette maquette, je me suis dit qu'il fallait que je construise quelque chose de simple à comprendre et facile à utiliser. L'objectif était que les utilisateurs puissent retrouver rapidement les informations importantes et effectuer les actions nécessaires sans se perdre. J'ai donc structuré la maquette de manière logique, en pensant à l'ordre dans lequel on aurait besoin des fonctionnalités. Chaque partie de l'interface a été pensée pour répondre à un besoin précis, que ce soit pour consulter une liste, voir un profil ou faire une action comme ajouter ou supprimer un employé. Le but, c'était vraiment de rendre l'utilisation fluide et efficace

Les différents pages pour le code :

Ajout_employe.php	Permet d'ajouter un nouvel employé dans la base de données via un formulaire.
ajout_equipement.php	Permet d'ajouter un nouvel équipement (EPI) à la base de données.
ajout_statut.php	Permet de créer un nouveau statut pour les employés, avec des limites d'EPI personnalisées.
config.php	Contient les informations de connexion à la base de données (hôte, utilisateur, mot de passe, nom de base).
index.php	Page d'accueil principale du système.
liste_employes.php	Affiche la liste de tous les employés avec options pour modifier, supprimer ou voir leur profil.
liste_statut.php	Affiche tous les statuts enregistrés et leurs limites d'EPI.
retraits.php	Gère les retraits d'EPI : enregistre la quantité et la date dans l'historique.
simulation.php	Simule un retrait ou un test RFID sans action réelle sur la base (utile pour tester le système).
statistique.php	Affiche des graphiques sur l'utilisation des EPI : par jour, par employé, par équipement.
stock_update.php	Met à jour les quantités en stock des équipements (après retrait ou réapprovisionnement).

stock.php	Affiche la quantité actuelle en stock pour chaque type d'EPI.
supprimer_employe.php	Supprime un employé de la base de données.
supprimer_statut.php	Supprime un statut de la base de données.
update_employe.php	Met à jour les informations d'un employé existant.
update_statut.php	Met à jour les limites ou le nom d'un statut.
voirprofil.php	Affiche les informations détaillées d'un employé (nom, statut, carte RFID, retraits d'EPI, alertes).

Explication d'une partie de code :

ajout_employe.php :

```
1 : *{ box-sizing: border-box; }
2 : html, body {
  margin: 0;
  padding: 0;
  height: 100%;
  width: 100%;
  font-family: 'Segoe UI', sans-serif;
 background: linear-gradient(135deg, #a0d8ef, #f2f2f2);
  background-size: 400% 400%;
  animation: gradientBG 10s ease infinite;
}
3 : @keyframes gradientBG {
  0% { background-position: 0% 50%; }
  50% { background-position: 100% 50%; }
  100% { background-position: 0% 50%; }
}
4 : h1 {
  text-align: center;
  font-size: 3em;
  color: #1a1a1a;
  margin-bottom: 40px;
  text-shadow: 2px 2px 4px rgba(0,0,0,0.2);
}
```

1 :

- Applique “`box-sizing: border-box`" à **tous les éléments** : cela fait en sorte que les marges et les bordures soient **incluses dans la taille totale** de l’élément.

2 :

- Réinitialise les marges et paddings du navigateur (`margin: 0`, `padding: 0`).
- Fixe la hauteur et largeur à 100% de la page.
- Change la police pour une police moderne.
- Définit un **fond en dégradé animé**.
- `background-size: 400% 400%` : rend l’animation plus fluide.
- `animation: gradientBG ...` applique l’effet défini plus bas.

3 :

- Conteneur principal de la page.
- Utilise Flexbox pour **organiser le contenu verticalement**.
- `space-between` laisse de l’espace entre les blocs.

4 :

- Titre principal centré, **très visible**.
- `text-shadow` : ajoute un **effet d’ombre** pour le relief.

Index.php :

```
<!DOCTYPE html>
```

```
<html lang="fr">
```

- Définit que le document est une page HTML, en langue française.

```
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1" />
 <title>Menu Principal - Gestion EPI</title>
```

- Le head contient les métadonnées : encodage en UTF-8, adaptabilité aux écrans mobiles, et le titre de la page affiché dans l'onglet du navigateur.

```
* {
  margin: 0; padding: 0; box-sizing: border-box;
}
```

- Réinitialise les marges et paddings pour éviter les incohérences entre navigateurs.

```
body, html {
  height: 100%;
  font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
  background: linear-gradient(135deg, #6a98f0, #a3c9ff);
  ...
}
```

- Donne un fond dégradé bleu et applique une police lisible sur toute la page.

```
header {
  display: flex;
  justify-content: space-between;
  ...
}
```

- Le header contient le titre à gauche et la date à droite. Il utilise un effet de flou (backdrop-filter) et un fond semi-transparent.

liste_statuts.php :

```
<!DOCTYPE html>
<html lang="fr">
```

```
<head>
 <meta charset="UTF-8">
 <title>Liste des Statuts</title>
 <style>
  /* Style simple et clair pour le tableau et la page */
 </style>
</head>
<body>
```

- Le doctype définit le type de document.
- La langue est fixée au français.
- Le titre s'affiche dans l'onglet du navigateur.
- Le CSS intégré stylise le corps, le tableau, les boutons et l'interaction au survol.

```
<a href="index.php" style="...">Retour sur la page Home</a>
```

- Un lien stylisé permettant de revenir à la page principale.

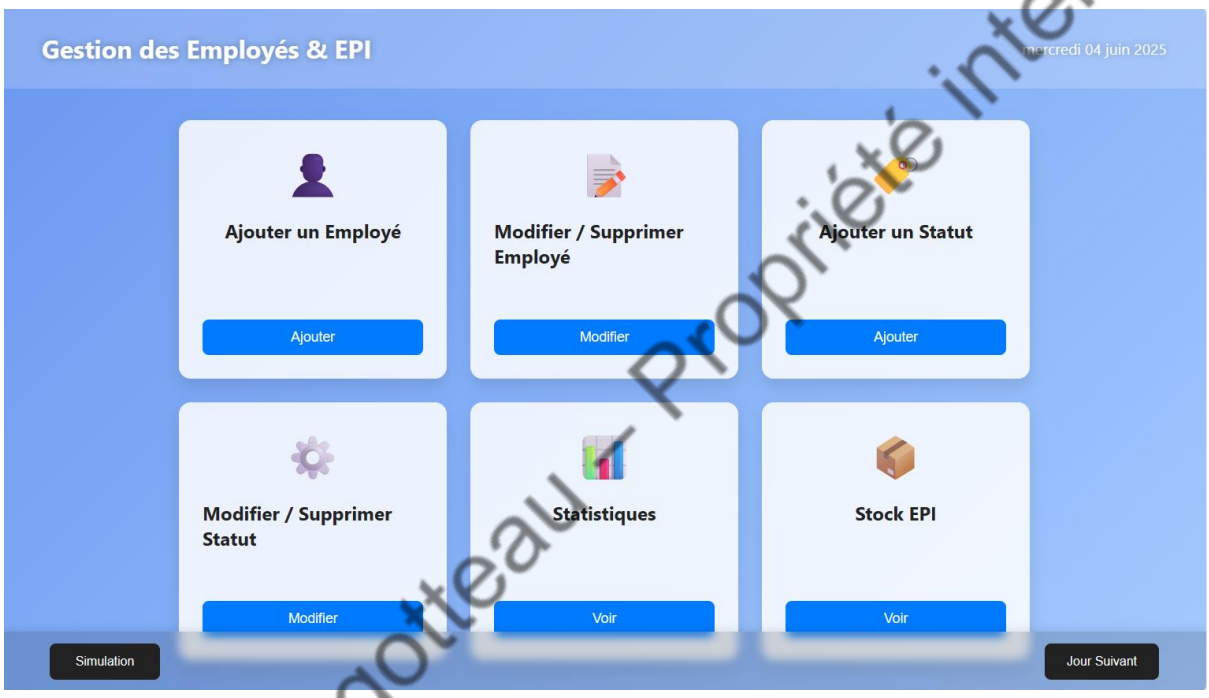
```
<h2>Liste des Statuts</h2>
```

- Titre centré pour indiquer clairement la fonction de la page.

```
<table>
 <thead>
  <tr>
   <th>ID</th>
   <th>Nom du Statut</th>
   <th>Max Gants</th>
   <th>Max Masques</th>
   <th>Max Boules</th>
   <th>Max Gilets</th>
   <th>Actions</th>
  </tr>
 </thead>
 <tbody>
```

- Entête du tableau pour afficher chaque colonne importante.
- Colonnes pour identifier le statut, afficher les limites d'EPI, et proposer des actions.

Image de l'index :



L'IHM développée dans ce projet a été pensée pour être claire, intuitive et fonctionnelle. Elle permet à l'utilisateur de naviguer facilement entre les différentes pages (accueil, historique, statistiques), de consulter rapidement les informations essentielles, et d'interagir avec les données grâce à des boutons et filtres simples d'utilisation.

Grâce à une organisation visuelle soignée, des couleurs cohérentes et des boutons bien identifiables, l'interface améliore l'expérience utilisateur et facilite la gestion des EPI. Les tableaux, formulaires et graphiques apportent une vue d'ensemble efficace et compréhensible, même pour un utilisateur non expert.

Cette IHM remplit donc son rôle principal : rendre l'application accessible, ergonomique et agréable à utiliser au quotidien.

Conclusion :

Ce projet nous a permis d'aborder les aspects écologique, économique et sociale liées à la conception d'un distributeur d'EPI. Nous avons appris à concevoir un projet technologique tout en travaillant en équipe, ce qui nous a permis de développer notre manière de communiquer entre les uns et les autres. Face aux différents problèmes rencontrés, nous avons su nous adapter sur certains points et avons pris conscience de nombreux éléments auxquels nous n'avions pas pensé au départ.

Cependant, plusieurs obstacles sont venus freiner notre progression. Nous avons rencontré des problèmes de gestion du temps, souvent dus à des bugs causés par des bibliothèques incompatibles. L'utilisation du module ESP8266 a également posé problème, notamment au niveau des ports COM. De plus, certains composants comme le moteur pas à pas Nema17 et le module L298N ont été difficiles à intégrer. Les ressources disponibles pour ces capteurs (Nema17 + L298N + ESP8266) étaient limitées ou contradictoires, ce qui nous a obligés à, regrouper différentes sources, et faire preuve de rigueur pour éviter les erreurs.

Nous devons encore progresser sur certains points, en particulier sur l'organisation et la gestion du temps, qui se sont révélées parfois insuffisantes face à l'ampleur du projet.

Malgré toutes ces difficultés, nous considérons ce projet comme une réussite. Il nous a permis de développer nos compétences techniques, de mieux comprendre les exigences du travail en groupe, et de prendre conscience de l'importance de la planification et des recherches dans la réalisation d'un projet.