



---

# RAPPORT DE PROJET

Création du système e-drivin

Réalisé par  
**PERERA Githendra**  
**GROLLEAU Lucas**  
**GROSHELLY Thibaut**

Sous la direction de **Palcy Le Moine**

---



# TABLE DES MATIÈRES

|  |           |
|--|-----------|
| <b>Remerciement.....</b>                       | <b>5</b>  |
| <b>Introduction.....</b>                       | <b>6</b>  |
| A – Les objectifs techniques.....              | 7         |
| B – Le cahier des charges.....                 | 8         |
| C – Vue globale du système.....                | 9         |
| D – Répartition des tâches.....                | 10        |
| E – Schéma de la base de donnée.....           | 11        |
| <b>I – Étudiant 1 : GROLLEAU Lucas.....</b>    | <b>12</b> |
| A – Planification.....                         | 12        |
| B – Diagramme de séquence.....                 | 13        |
| C – Diagramme de cas d'utilisation.....        | 14        |
| D – Logiciel utilisés.....                     | 15        |
| E – Programmes.....                            | 16        |
| F – Conclusion.....                            | 35        |
| <b>II – Étudiant 2 : PERERA Githendra.....</b> | <b>36</b> |
| A – Planification.....                         | 36        |
| B – Diagramme de séquence.....                 | 37        |

|   |           |
|---|-----------|
| C – Diagramme de cas d'utilisation.....                                       | 38        |
| D – Logiciel utilisés.....  | 39        |
| E – Matériel principal de ma partie.....                                      | 40        |
| F – Diagramme de classe de l'IHM.....   | 41        |
| G – Programme IHM.....  | 42        |
| H – Diagramme de classe pour l'affichage des commandes des clients.....       | 56        |
| J – Programme de l'affichage des commandes des clients.....                   | 57        |
| K – Conclusion.....   | 64        |
| <b>III – Étudiant 3 : GROSHENY Thibaut.....</b>                               | <b>65</b> |
| A – Module de facturation.....  | 65        |
| B - Planification.....  | 66        |
| C – Diagramme de séquence.....  | 66        |
| D - Diagramme déploiement.....  | 67        |
| E - Diagramme cas d'utilisation.....  | 67        |
| F - Présentation de ce Module.....  | 68        |
| G - Présentation du matériel et du logiciel utilisé.....                      | 69        |
| H - Fonctionnement du lecteur code barre avec l'infrarouge.....               | 70        |
| I - Liste des différents codes barres et explication du code barre EAN13..... | 71        |
| J - Organigramme des étapes du programme.....                                 | 72        |

|   |            |
|---|------------|
| K - Connexion à la Base de données.....                     | 73         |
| L - Découpage du code barre.....                            | 74         |
| M - Récupération des informations de la BDD.....            | 76         |
| N - Récupération des informations client.....               | 79         |
| O - Enregistrement de la facture.....                       | 80         |
| P - Impression de la facture.....                           | 81         |
| <b>IV – Étudiant 3 : GROSHENY Thibaut Partie Bonus.....</b> | <b>82</b>  |
| A - Module Paiement.....                                    | 82         |
| B – Planification.....                                      | 83         |
| C - Diagramme séquence.....                                 | 83         |
| D - Diagramme de déploiement.....                           | 84         |
| E - Diagramme cas d'utilisation.....                        | 84         |
| F - Présentation de ce Module.....                          | 85         |
| G - Présentation du matériel et du logiciel utilisé.....    | 86         |
| H - Câblage du module.....                                  | 88         |
| I - Organigramme des étapes du programme.....               | 89         |
| J - Écriture sur la carte ou le badge.....                  | 90         |
| K - Lecture de la carte ou du badge.....                    | 92         |
| <b>Étudiant 1: ANNEXE.....</b>                              | <b>94</b>  |
| <b>Étudiant 2: ANNEXE.....</b>                              | <b>122</b> |
| <b>Étudiant 3: ANNEXE.....</b>                              | <b>145</b> |

# REMERCIEMENTS

Nous nous permettons tout d'abord de remercier Madame Janick PALCY LE MOINE de nous avoir aidés pour le projet ainsi que pour chacune de nos parties personnelles.

Enfin, nous souhaitons exprimer notre gratitude envers Monsieur OUHAMMOU LAHOUCINE pour ses précieux conseils.

# INTRODUCTION

Le système e-drivin est un service proposé par les supermarchés pour permettre à leurs clients de commander en ligne les produits qu'ils souhaitent acheter et de venir les récupérer directement en magasin sans avoir à perdre du temps à chercher les produits et à faire la queue en caisse.

Le client peut accéder au service e-drivin via le site web du supermarché. Il peut ainsi parcourir les différentes catégories de produits proposés et ajouter les articles de son choix à son panier virtuel.

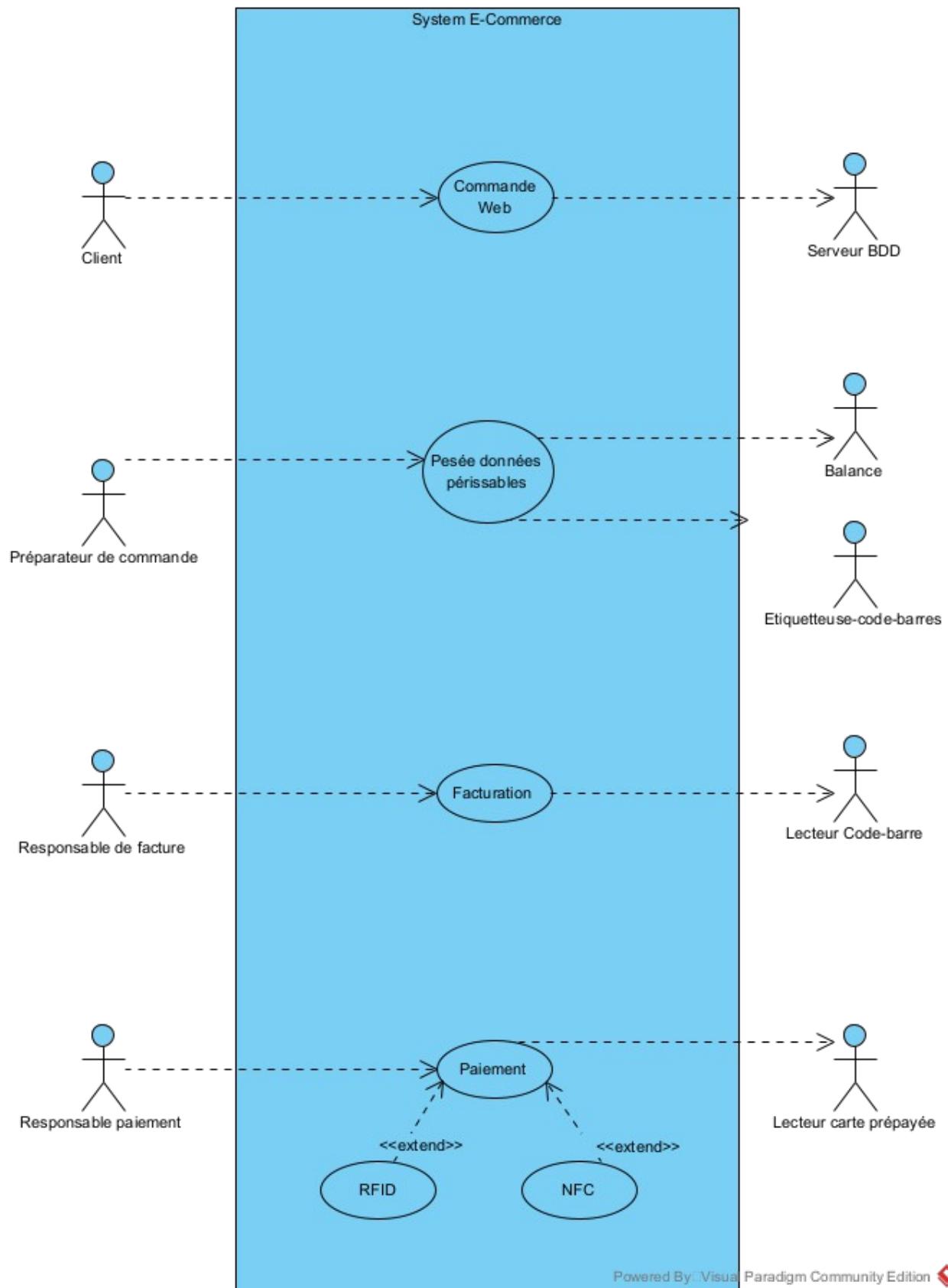
Une fois que la commande est passée, le client reçoit une confirmation de commande avec un numéro de commande et une heure de récupération estimée. À l'heure prévue, le client se rend directement au point de retrait e-drivin situé à l'extérieur du magasin, où un employé du supermarché lui remet ses courses déjà préparées.

Le paiement peut être effectué en ligne au moment de la commande ou directement sur place lors de la récupération des produits. Le service e-drivin est de plus en plus populaire auprès des clients car il leur permet de gagner du temps tout en ayant un accès facile et rapide aux produits dont ils ont besoin.

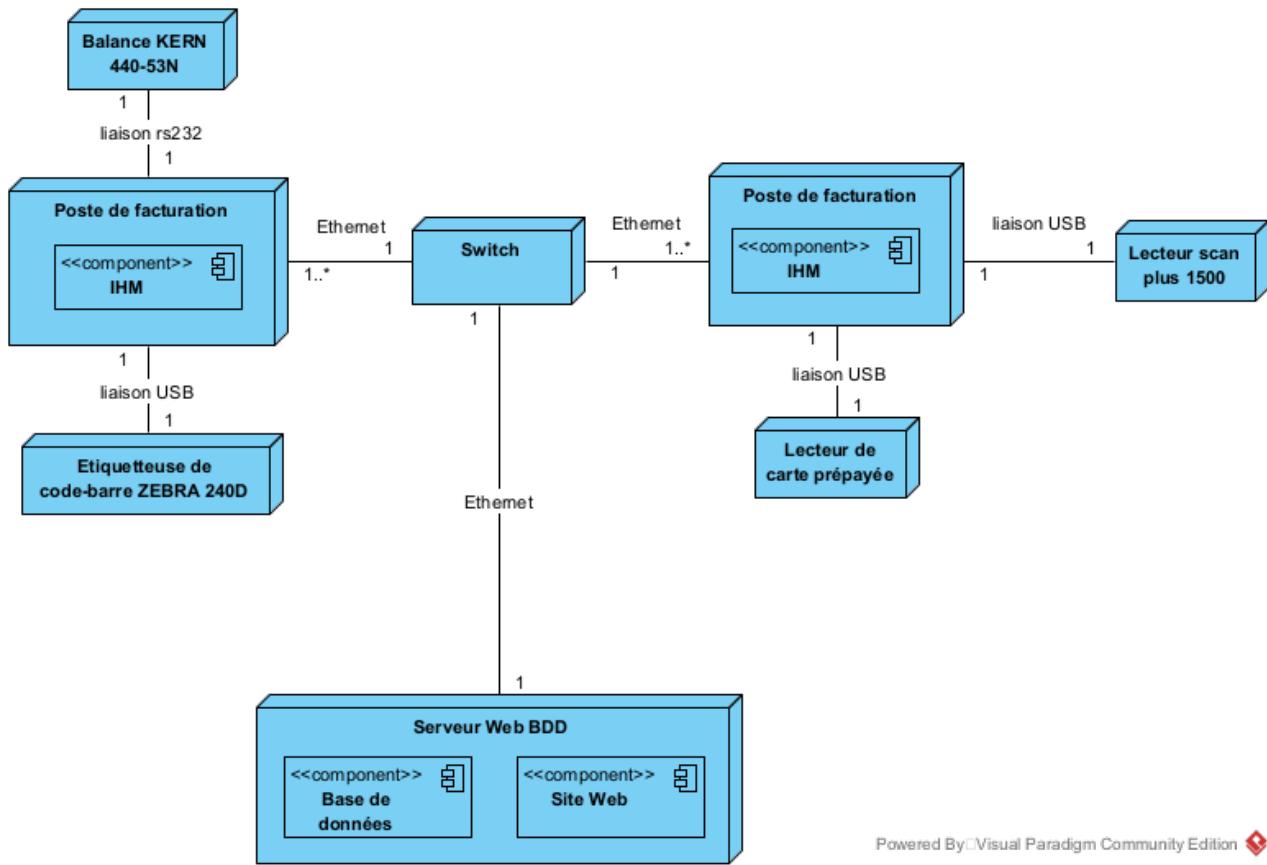
## A – Les objectifs techniques

- Développer une plateforme web robuste et sécurisée qui permet aux clients de passer leurs commandes de produits en ligne.
- Concevoir une interface utilisateur intuitive pour faciliter la recherche de produits, la sélection et la confirmation des commandes.
- Établir un système de gestion des stocks en temps réel pour assurer la disponibilité des produits commandés et éviter les erreurs de commande.
- Mettre en place un système de paiement qui permettra aux clients de régler leurs commandes au moment de la récupération des produits commandés.

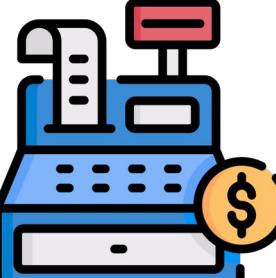
## B – Le cahier des charges



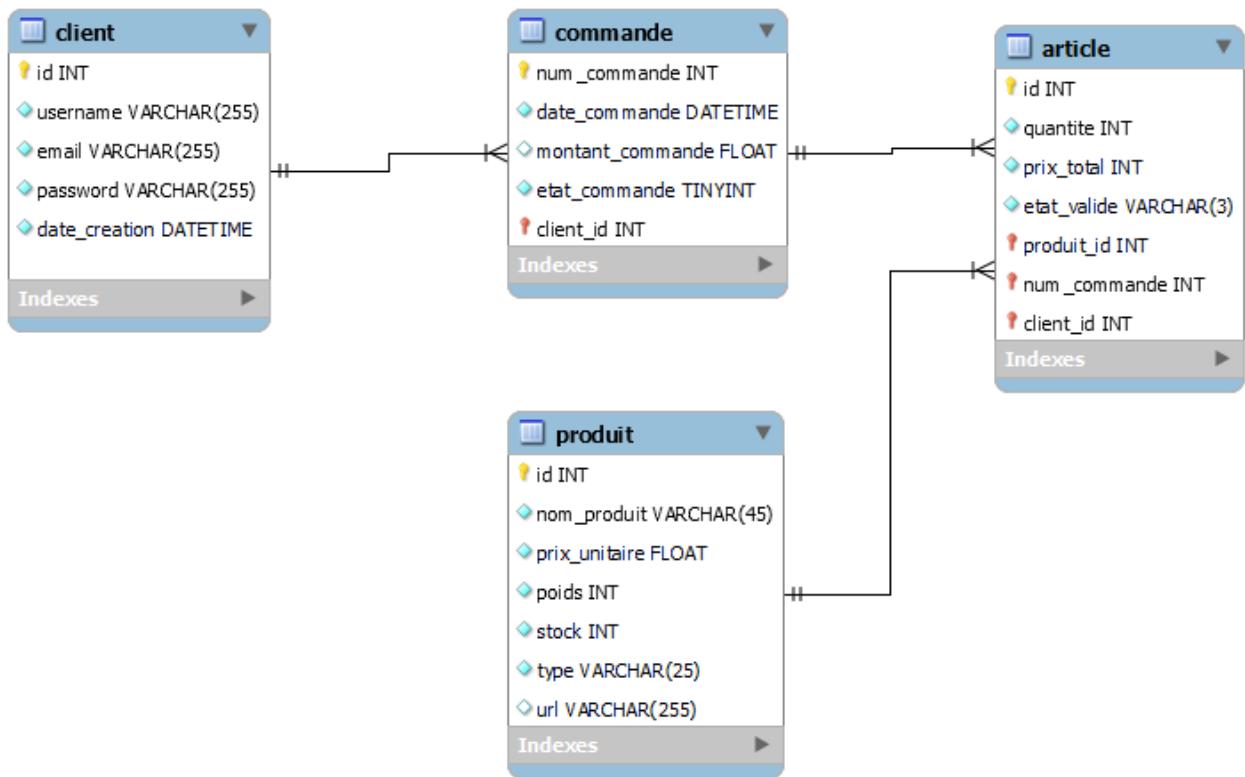
## C – Vue globale du système



## D – Répartition des tâches

|   |  |
|---|--|
|    | <p>Étudiant 1: GROLLEAU Lucas</p> <ul style="list-style-type: none"><li>- Codage module logiciel de gestion de la base de données</li><li>- Codage du module logiciel de gestion des pages web dynamique</li></ul>   |
|   | <p>Étudiant 2: PERERA Githendra</p> <ul style="list-style-type: none"><li>- Conception du module logiciel de gestion de l'imprimante et de la balance</li><li>- Impression d'un ticket avec un code barre</li></ul>  |
|  | <p>Étudiant 3: GROSHENY Thibaut</p> <ul style="list-style-type: none"><li>- Conception du module logiciel de gestion du lecteur code barre.</li><li>- Codage du module logiciel d'impression d'une facture papier</li><li>- Mise en place du lecteur carte à puce USB.</li><li>- Mise en place d'une impression de ticket papier</li></ul> |

## E – Schéma de la base de donnée



La table "produit" sert à stocker les produits vendus au magasin.

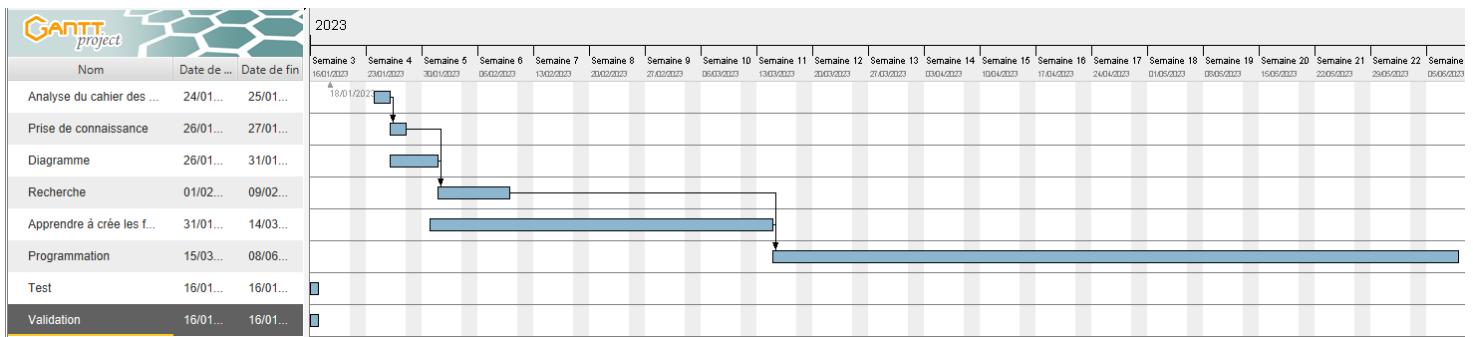
La table "commande" sert à archiver les commandes de chaque client.

La table "article" sert à stocker et à archiver les articles commandés ainsi que leurs quantités pour chaque client.

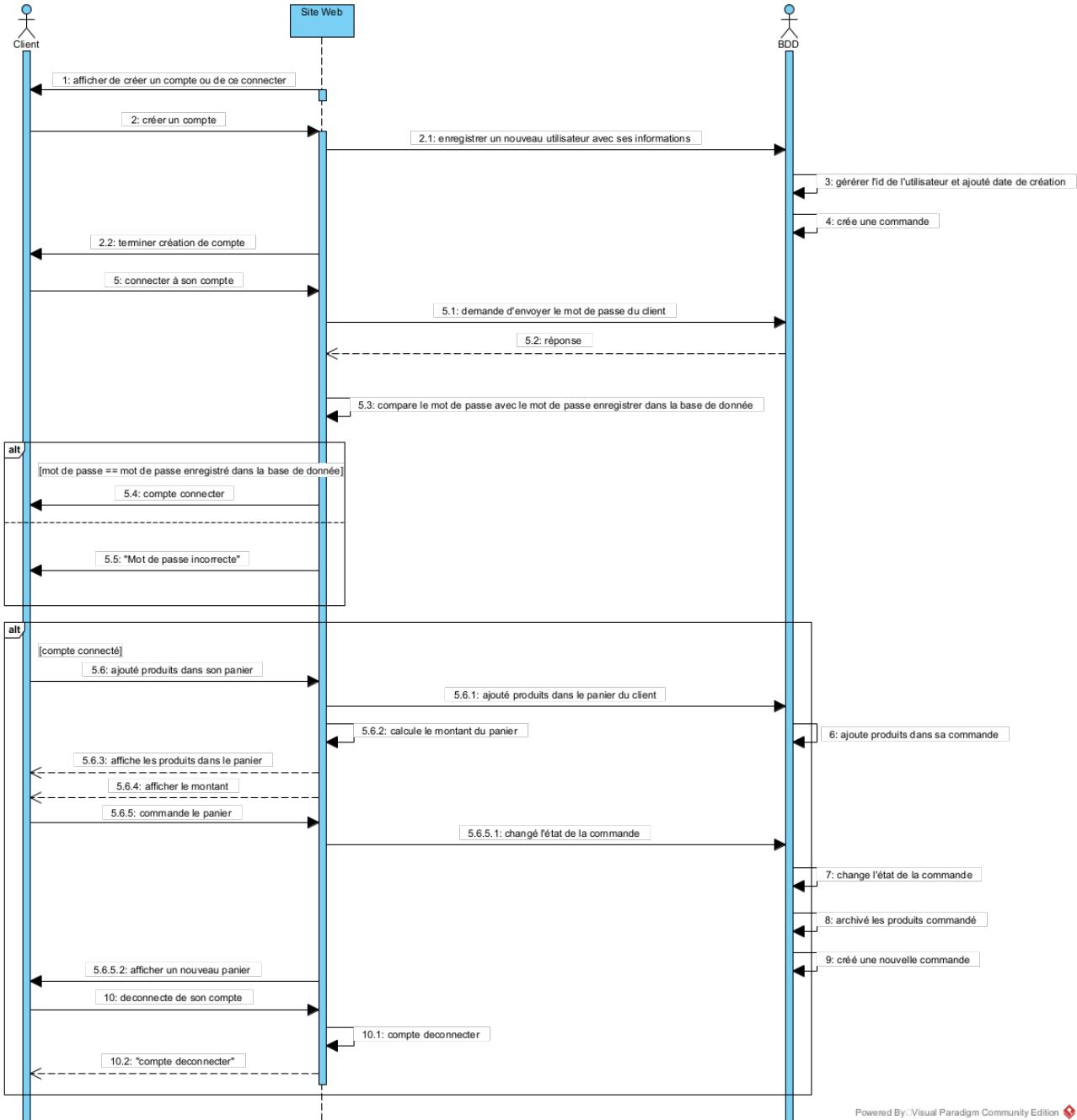
La table "client" sert à stocker les informations relatives aux clients.

# I - Étudiant 1: GROLLEAU Lucas

## A – Planification :

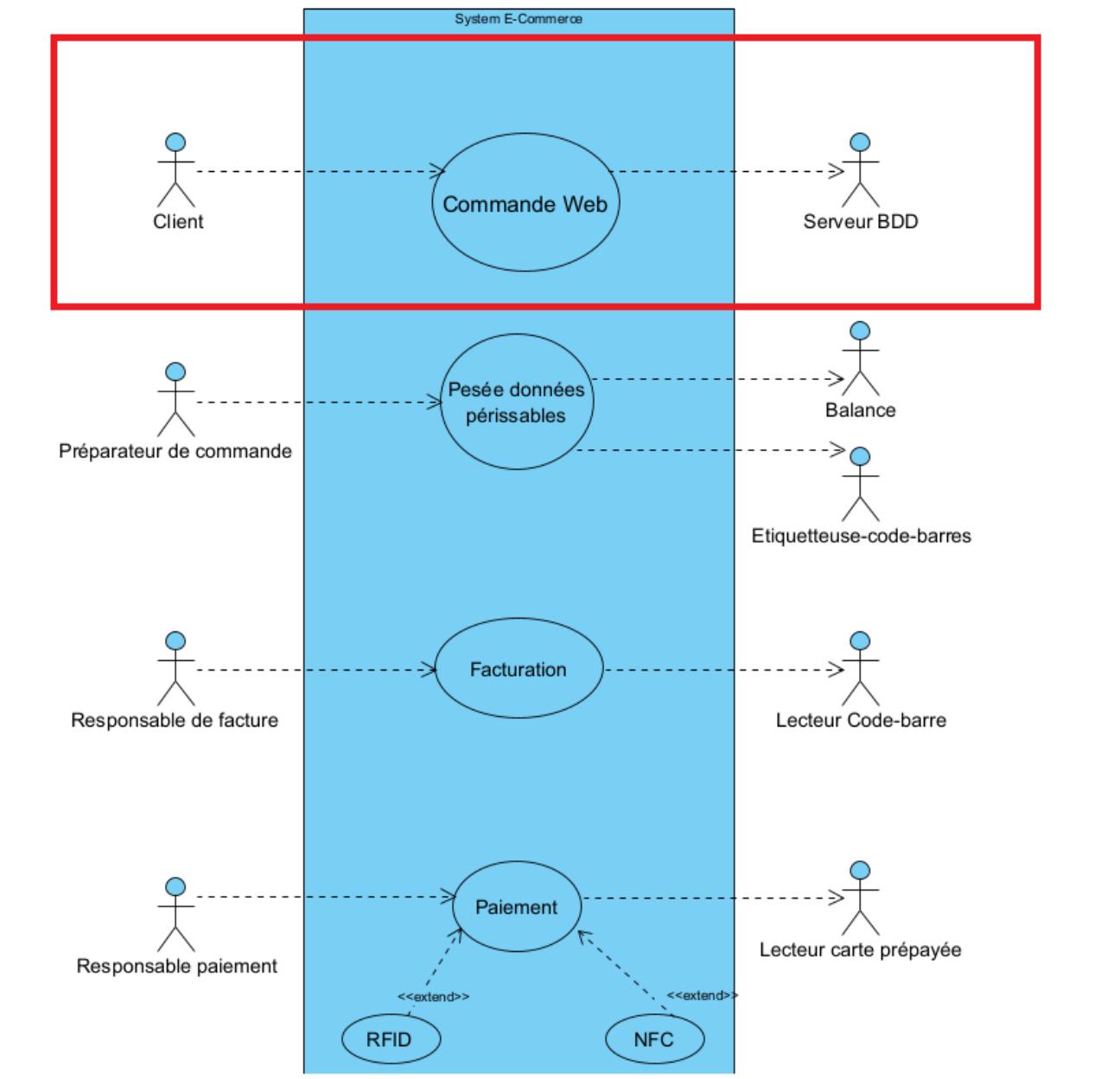


## B – Diagramme de séquence :



Ce diagramme de séquence représente le fonctionnement général de ma partie. Le but est de donner un accès en ligne au futurs clients pour qu'ils puissent commander en ligne leurs courses et pouvoir les récupérés plus tard.

## C – Diagramme de cas d'utilisation :



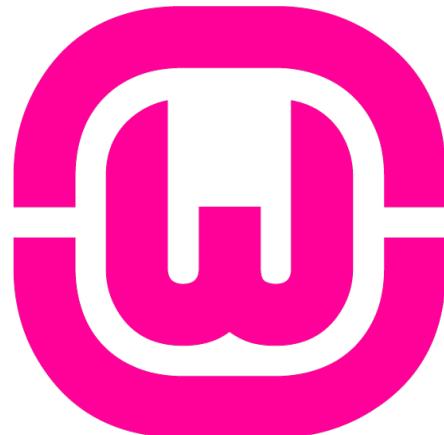
Ce diagramme de cas d'utilisation représente le fonctionnement du système en général.

Ma partie est celle des composants dans l'encadré rouge. C'est là que je gère la connexion des clients pour qu'ils puissent ajouter des produits directement en ligne dans leur panier, puis leur donner la possibilité de pouvoir commander leur panier bien remplis.

## D – Logiciels utilisés :



Sublime Text est un éditeur de texte avancé et populaire utilisé par de nombreux développeurs et programmeurs. Il est conçu pour offrir une expérience de modification de texte rapide et fluide, avec des fonctionnalités puissantes et une interface utilisateur élégante.



WampServer est un logiciel gratuit qui permet de mettre en place un environnement de développement web sur un ordinateur local. Il fournit une plateforme intégrée comprenant le serveur web Apache, le système de gestion de bases de données MySQL/MariaDB, ainsi que le langage de script PHP.



PHPMyAdmin est une application web open-source utilisée pour gérer des bases de données MySQL. C'est un outil très populaire et largement utilisé par les développeurs et les administrateurs de bases de données.



MySQL Workbench est un logiciel de gestion et d'administration de bases de données MySQL. Via une interface graphique intuitive, il permet de créer, modifier ou supprimer des tables, des comptes utilisateurs, et d'effectuer toutes les opérations inhérentes à la gestion d'une base de données. Pour ce faire, il doit être connecté à un serveur MySQL.

## E – Programmes:

```
1 <?php
2 require_once('config.php');
3
4 function register_user($email, $password) {
5     global $db;
6
7     // Vérifier si l'utilisateur existe déjà
8     $query = "SELECT * FROM users WHERE email = '$email'";
9     $result = mysqli_query($db, $query);
10    if (mysqli_num_rows($result) > 0) {
11        return 'Cet email est déjà utilisé';
12    }
13
14    // Hacher le mot de passe
15    $hashed_password = password_hash($password, PASSWORD_DEFAULT);
16
17    // Insérer l'utilisateur dans la base de données
18    $query = "INSERT INTO users (email, password, created_at) VALUES ('$email', '$hashed_password', NOW())";
19    mysqli_query($db, $query);
20
21    return 'Compte créé avec succès';
22 }
23
24 function login_user($email, $password) {
25     global $db;
26
27     // Récupérer l'utilisateur correspondant à l'email
28     $query = "SELECT * FROM users WHERE email = '$email'";
29     $result = mysqli_query($db, $query);
30     if (mysqli_num_rows($result) == 0) {
31         return 'Email ou mot de passe incorrect';
32     }
33     $user = mysqli_fetch_assoc($result);
34
35     // Vérifier le mot de passe
36     if (!password_verify($password, $user['password'])) {
37         return 'Email ou mot de passe incorrect';
38     }
39
40     // Démarrer la session et stocker l'ID de l'utilisateur
41     session_start();
42     $_SESSION['user_id'] = $user['id'];
43
44     return 'Connexion réussie';
45 }
46
47 function logout_user() {
48     session_start();
49     unset($_SESSION['user_id']);
50     session_destroy();
51     header('Location: index.php');
52     exit();
53 }
54
55 function is_user_logged_in() {
56     session_start();
57     return isset($_SESSION['user_id']);
58 }
59
60 function get_logged_in_user() {
61     global $db;
62
63     session_start();
64
65     if (!isset($_SESSION['user_id'])) {
66         return false;
67     }
68
69     $user_id = $_SESSION['user_id'];
70     $query = "SELECT * FROM users WHERE id = $user_id";
71     $result = mysqli_query($db, $query);
72     $user = mysqli_fetch_assoc($result);
73
74     return $user;
75 }
76 ?>
```

Ce code contient des fonctions liées à l'inscription, la connexion, la déconnexion et la gestion des utilisateurs dans un système d'authentification.

- La fonction `register_user` permet d'enregistrer un nouvel utilisateur dans une base de données. Elle vérifie d'abord si l'email existe déjà, puis elle hache le mot de passe avant de l'insérer dans la base de données.
- La fonction `login_user` permet à un utilisateur de se connecter en vérifiant l'existence de l'email dans la base de données et en comparant le mot de passe haché avec celui saisi. Si la connexion réussit, elle démarre une session et stocke l'ID de l'utilisateur.
- La fonction `logout_user` permet de déconnecter l'utilisateur en détruisant la session et en redirigeant vers la page d'accueil.
- La fonction `is_user_logged_in` vérifie si un utilisateur est connecté en vérifiant la présence de l'ID utilisateur dans la session.
- La fonction `get_logged_in_user` récupère les informations de l'utilisateur connecté en utilisant l'ID stocké dans la session et les renvoie.

#### Config.php:

```
1 <?php
2 $host = 'localhost';
3 $username = 'root';
4 $password = '';
5 $database = 'site_e-commerce';
6 $db = mysqli_connect($host, $username, $password, $database);
7 if (!$db) {
8     die('Erreur de connexion: ' . mysqli_connect_error());
9 }
10
11 // Vérifier si l'utilisateur est connecté
12 session_start();
13 $userLoggedIn = false;
14 if(isset($_SESSION["loggedin"]) && $_SESSION["loggedin"] === true){
15     $userLoggedIn = $_SESSION["id"];
16 }
17 ?>
```

Ce code établit une connexion à une base de données MySQL et vérifie si un utilisateur est connecté en utilisant les sessions PHP. Voici un résumé des actions effectuées :

1. Définition des informations de connexion à la base de données (hôte, nom d'utilisateur, mot de passe, nom de la base de données).
2. Établissement d'une connexion à la base de données en utilisant les informations de connexion fournies.
3. Vérification de la connexion à la base de données et affichage d'un message d'erreur en cas d'échec.
4. Démarrage de la session PHP.
5. Initialisation d'une variable `$userLoggedIn` à `false`.
6. Vérification si la variable de session "loggedin" est définie et égale à `true`.
7. Si la condition est vraie, la variable `$userLoggedIn` est mise à jour avec la valeur de la variable de session "id" de l'utilisateur connecté.

Ce code permet donc d'établir une connexion à la base de données et de vérifier l'état de connexion de l'utilisateur.

## Register.php:

```
1 <?php
2 require_once('includes/config.php');
3
4 if($_SERVER['REQUEST_METHOD'] == 'POST') {
5     $username = trim($_POST['username']);
6     $email = trim($_POST['email']);
7     $password = trim($_POST['password']);
8     $confirm_password = trim($_POST['confirm_password']);
9
10    $errors = [];
11
12    if(empty($username)) {
13        $errors[] = 'Le champ "Nom d\'utilisateur" est obligatoire';
14    }
15
16    if(empty($email)) {
17        $errors[] = 'Le champ "Email" est obligatoire';
18    } else {
19        if(!filter_var($email, FILTER_VALIDATE_EMAIL)) {
20            $errors[] = 'Le format de l\'email n\'est pas valide';
21        }
22    }
23
24    if(empty($password)) {
25        $errors[] = 'Le champ "Mot de passe" est obligatoire';
26    } else {
27        if(strlen($password) < 6) {
28            $errors[] = 'Le mot de passe doit contenir au moins 6 caractères';
29        }
30        if($password !== $confirm_password) {
31            $errors[] = 'Les mots de passe ne correspondent pas';
32        }
33    }
34
35    if(empty($errors)) {
36        $hashed_password = password_hash($password, PASSWORD_DEFAULT); // Hashage du mot de passe
37        date_default_timezone_set('UTC');
38        $date_creation = date('Y-m-d H:i:s');
39
40        $stmt = $db->prepare("INSERT INTO client (username, email, password, date_creation) VALUES (?, ?, ?, ?)");
41        if (!$stmt) {
42            die('Erreur de préparation de la requête : ' . $db->error);
43        }
44        $stmt->bind_param("ssss", $username, $email, $hashed_password, $date_creation);
45        if (!$stmt->execute()) {
46            die('Erreur d\'exécution de la requête : ' . $stmt->error);
47        }
48        // Récupérer l'id du client inséré
49        $client_id = $stmt->insert_id;
50
51        // Créer une nouvelle commande
52        $date_commande = date('Y-m-d H:i:s');
53        $montant_commande = 0;
54        $etat_commande = 0;
55
56        $stmt = $db->prepare("INSERT INTO commande (date_commande, montant_commande, etat_commande, client_id) VALUES (?, ?, ?, ?)");
57        if (!$stmt) {
58            die('Erreur de préparation de la requête : ' . $db->error);
59        }
60        $stmt->bind_param("sdii", $date_commande, $montant_commande, $etat_commande, $client_id);
61        if (!$stmt->execute()) {
62            die('Erreur d\'exécution de la requête : ' . $stmt->error);
63        }
64
65        header('Location: login.php');
66        exit;
67    }
68 }
69 ?>
```

Ce code est une page PHP qui permet à un utilisateur de créer un compte sur un site web. Voici un résumé de ce que fait le code :

1. Il inclut le fichier de configuration (`config.php`) qui contient les paramètres de connexion à la base de données.
2. Vérifie si la méthode de requête est POST, ce qui indique que le formulaire a été soumis.
3. Récupère les valeurs des champs du formulaire (`username`, `email`, `password`, `confirm_password`) et les nettoie en enlevant les espaces superflus.
4. Effectue des validations sur les champs pour s'assurer qu'ils ne sont pas vides et ont le bon format. Les erreurs sont stockées dans un tableau `$errors`.
5. Si aucune erreur n'est détectée, le mot de passe est hashé à l'aide de la fonction `password_hash()` et la date de création du compte est enregistrée.
6. Une requête SQL est préparée pour insérer les données du nouvel utilisateur dans la table "client" de la base de données.
7. Une fois l'utilisateur inséré avec succès, l'ID du client est récupéré.
8. Une nouvelle commande est créée en insérant les informations de la commande dans la table "commande" de la base de données.
9. Enfin, si tout s'est déroulé avec succès, l'utilisateur est redirigé vers la page de connexion (`login.php`).

Le reste du code est du code HTML qui affiche le formulaire d'inscription avec les messages d'erreur appropriés s'il y a des erreurs de validation.

### Login.php:

Ce code est une page de connexion en PHP avec un formulaire permettant aux utilisateurs de se connecter. Voici un résumé des principales fonctionnalités du code :

- Le code commence par l'inclusion d'un fichier de configuration et de connexion à la base de données.
- Il initialise des variables pour stocker les informations d'identification de l'utilisateur (nom d'utilisateur et mot de passe) ainsi que des variables d'erreur associées.
- Lorsque le formulaire de connexion est soumis (méthode POST), le code effectue les validations nécessaires sur le nom d'utilisateur et le mot de passe saisis.
- Si les validations réussissent, une requête SELECT est préparée pour récupérer l'utilisateur correspondant aux informations d'identification fournies.
- Si l'utilisateur est trouvé dans la base de données et que le mot de passe corre-

spond, une session est démarrée et les informations de l'utilisateur sont stockées dans des variables de session.

- L'utilisateur est ensuite redirigé vers la page d'accueil du tableau de bord.
- Si des erreurs se produisent lors de la validation ou de l'exécution de la requête, des messages d'erreur appropriés sont affichés.
- Le code HTML contient le formulaire de connexion avec des champs pour le nom d'utilisateur et le mot de passe, ainsi que des messages d'erreur associés.
- Le code JavaScript inclus permet de basculer la visibilité du champ de mot de passe entre texte masqué et texte clair.
- Il y a également du code CSS pour le style et la mise en forme de la page.

En résumé, ce code gère le processus de connexion des utilisateurs en vérifiant leurs informations d'identification et en démarrant une session si elles sont valides.

## Index.php :

Ce code est une page web écrite en PHP. Voici un résumé des principales fonctionnalités :

### 1. Vérification de l'authentification de l'utilisateur :

- Si l'utilisateur n'est pas connecté (la variable de session 'username' n'est pas définie), il est redirigé vers la page de connexion.
- Sinon, le contenu de la page est affiché.

### 2. Ajout d'un produit au panier :

- Si le formulaire a été soumis avec les champs 'id' et 'quantite' définis, le produit correspondant à l'ID est récupéré depuis la base de données.
- Si le produit est périssable, la quantité est convertie en kilogrammes.
- Le produit est ajouté au panier, qui est stocké dans la variable de session 'panier'.
- L'utilisateur est ensuite redirigé vers la page du panier.

### 3. Récupération des produits depuis la base de données :

- Les produits sont récupérés depuis la table "produit" de la base de données.

Le reste du code concerne l'affichage des produits sur la page HTML :

- Le code HTML contient un en-tête (header) avec un logo, un titre et une barre de navigation.
- Dans la section principale (main), chaque produit est affiché dans une boucle foreach.
- Pour chaque produit, une image, un nom, un prix, un formulaire d'ajout au panier et un bouton sont affichés.

- Le formulaire envoie les données (ID du produit et quantité) à la page 'add.php' si l'utilisateur est connecté, sinon à la page 'login.php'.
- Un script JavaScript est inclus pour mettre à jour le prix total en fonction de la quantité sélectionnée pour chaque produit.
- Le pied de page (footer) affiche un message de copyright.
- Des scripts JavaScript externes sont inclus pour la manipulation du DOM et des fonctionnalités supplémentaires.

```

1 <?php
2 // Inclure le fichier de configuration et de connexion à la base de données
3 require_once('includes/config.php');
4
5 // Initialiser les variables pour stocker les informations d'identification de l'utilisateur
6 $username = "";
7 $password = "";
8 $username_err = "";
9 $password_err = "";
10
11 // Traitement du formulaire de connexion lors de la soumission
12 if ($_SERVER["REQUEST_METHOD"] == "POST") {
13     // Valider le nom d'utilisateur
14     if (!isset($_POST["username"]) || strlen(trim($_POST["username"])) == 0) {
15         $username_err = "Veuillez entrer votre nom d'utilisateur.";
16     } else {
17         $username = trim($_POST["username"]);
18     }
19
20     // Valider le mot de passe
21     if (!isset($_POST["password"]) || strlen(trim($_POST["password"])) == 0) {
22         $password_err = "Veuillez entrer votre mot de passe.";
23     } else {
24         $password = trim($_POST["password"]);
25     }
26
27     // Vérifier les erreurs de saisie avant de continuer
28     if (empty($username_err) && empty($password_err)) {
29         // Préparer une requête SELECT pour récupérer l'utilisateur correspondant aux informations d'identification fournies
30         $sql = "SELECT id, username, password FROM client WHERE username = ?";
31
32         if ($stmt = $db->prepare($sql)) {
33             // Définir les paramètres et lier les variables à la requête préparée en tant que paramètres
34             $param_username = $username;
35             $stmt->bind_param("s", $param_username);
36
37             // Exécuter la requête préparée
38             if ($stmt->execute()) {
39                 // Stocker le résultat
40                 $stmt->store_result();
41
42                 // Vérifier si le nom d'utilisateur existe, si oui alors vérifier le mot de passe
43                 if ($stmt->num_rows == 1) {
44                     // Lier les résultats de la requête à des variables
45                     $stmt->bind_result($id, $username, $password_db);
46                     if ($stmt->fetch()) {
47                         // Vérifier si le mot de passe saisi correspond au mot de passe dans la base de données
48                         if (password_verify($password, $password_db)) {
49                             // Le mot de passe est correct, alors commencer une session
50                             session_start();
51
52                             // Stocker les données de l'utilisateur dans des variables de session
53                             $_SESSION["loggedin"] = true;
54                             $_SESSION["id"] = $id;
55                             $_SESSION["username"] = $username;
56
57                             // Rediriger l'utilisateur vers la page de tableau de bord
58                             echo "<script>window.location.replace('index.php');</script>";
59                         } else {
56                             // Afficher un message d'erreur si le mot de passe est incorrect
57                             $password_err = "Le mot de passe que vous avez entré n'est pas valide.";
58                         }
59                     }
60                 } else {
61                     // Afficher un message d'erreur si le nom d'utilisateur n'existe pas
62                     $username_err = "Aucun compte trouvé avec ce nom d'utilisateur.";
63                 }
64             } else {
65                 // Afficher un message d'erreur si le nom d'utilisateur n'existe pas
66                 $username_err = "Aucun compte trouvé avec ce nom d'utilisateur.";
67             }
68         } else {
69             echo "Oops! Quelque chose s'est mal passé. Veuillez réessayer plus tard.";
70         }
71     }
72
73     // Fermer la déclaration préparée
74     $stmt->close();
75 }
76
77 }
78
79 // Fermer la connexion à la base de données
80 $db->close();
81 }
82 ?>

```

## Panier.php :

Ce code est une page PHP qui affiche le contenu du panier d'un utilisateur connecté sur un site de commerce électronique. Voici un résumé des fonctionnalités principales :

1. Établissement de la connexion à la base de données MySQL à l'aide des informations de connexion spécifiées.
2. Définition d'une fonction appelée `getProduitById` qui récupère les informations d'un produit en fonction de son ID.
3. Vérification si l'utilisateur est connecté. Si ce n'est pas le cas, il est redirigé vers la page de connexion.
4. Récupération de l'ID de l'utilisateur connecté à partir de la session.
5. Récupération des informations du panier de l'utilisateur à partir de la base de données. Cela inclut le numéro de commande, la date de commande, l'ID du produit, le nom du produit, le prix unitaire, la quantité, l'URL de l'image du produit et le prix total (calculé en multipliant le prix unitaire par la quantité).
6. Calcul du montant total de la commande en additionnant les prix totaux de tous les produits du panier.
7. Mise à jour du montant total de la commande dans la table de commandes de la base de données.
8. Affichage des informations du panier dans un tableau HTML, y compris l'image du produit, le nom du produit, le prix unitaire, la quantité, le prix total et un lien pour supprimer l'article du panier.
9. Affichage du montant total de la commande en bas du tableau.
10. Un bouton "Confirmer Commande" est affiché, qui, lorsqu'il est cliqué, déclenche une redirection vers une page de confirmation après une pause de 8 secondes.
11. La page HTML est stylisée avec des classes CSS et utilise également des fichiers CSS et JavaScript externes pour des fonctionnalités supplémentaires (animations, etc.).

```

try {
    $pdo = new PDO($dsn, $username, $password);
} catch (PDOException $e) {
    echo 'Connexion échouée : ' . $e->getMessage();
}

function getProduitById($produit_id) {
    global $pdo;
    $stmt = $pdo->prepare("SELECT * FROM produit WHERE id = :produit_id");
    $stmt->bindParam(":produit_id", $produit_id, PDO::PARAM_INT);
    $stmt->execute();
    return $stmt->fetch(PDO::FETCH_ASSOC);
}

// Vérifie si l'utilisateur est connecté, sinon redirige vers la page de connexion
if(!isset($_SESSION["loggedin"]) || $_SESSION["loggedin"] != true){
    header("location: login.php");
    exit;
}

// Récupérer l'id de l'utilisateur connecté
$client_id = $_SESSION["id"];

// Récupérer les informations du panier
$sql = "SELECT c.num_commande, c.date_commande, p.id AS produit_id, p.nom_produit, p.prix_unitaire, a.quantite, p.url, (p.prix_unitaire * a.quantite) AS prix_total
        FROM commande c
        JOIN article a ON c.num_commande = a.num_commande AND c.client_id = a.client_id
        JOIN produit p ON a.produit_id = p.id
        WHERE c.client_id = :client_id AND c.etat_commande = 0";
if($stmt = $pdo->prepare($sql)){
    $stmt->bindParam(":client_id", $client_id, PDO::PARAM_INT);

    if($stmt->execute()){
        $rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
    } else{
        echo "Oops! Something went wrong. Please try again later.";
    }
}

// Calculer le montant total de la commande
$total = 0;
foreach($rows as $row){
    $prix_total = $row["prix_total"];
    $total += $prix_total;
}

// Envoyer le montant total à la table commande
$sql = "UPDATE commande SET montant_commande = :total WHERE client_id = :client_id AND etat_commande = 0";
if($stmt = $pdo->prepare($sql)){
    $stmt->bindParam(":total", $total, PDO::PARAM_INT);
    $stmt->bindParam(":client_id", $client_id, PDO::PARAM_INT);
    if(!$stmt->execute()){
        echo "Oops! Something went wrong. Please try again later.";
    }
}
?>

```

```

<tbody>
    <?php
        $total = 0;
        foreach($rows as $row){
            $prix_total = $row["prix_total"];
            $total += $prix_total;
            $produit = getProduitById($row["produit_id"]);
        ?>
        <tr>
            <td style="text-align: center;"><img src=<?php echo $produit["url"]; ?> width="50" height="50" align-items="center"></td>
            <td><?php echo $produit["nom_produit"]; ?></td>
            <td><?php echo $produit["prix_unitaire"]."€"; ?></td>
            <td><?php echo $row["quantite"]; ?></td>
            <td><?php echo $prix_total."€"; ?></td>
            <td style="text-align: center;"><a href="supprimer_article.php"></a></td>
        </tr>
    <?php } ?>
    <thead>
        <tr>
            <td bgcolor="white" colspan="4"></td>
            <td colspan="1"><strong>Total</strong></td>
            <td><strong><?php echo $total."€"; ?></strong></td>
        </tr>
    </thead>
</tbody>
</table>
<div class="button-container" align="right">
    <button class="order" onclick="setTimeout(function(){ window.location.href = 'confirmation.php'; }, 8000);">
        <span class="default">Confirmer Commande</span>
        <span class="success">Commande effectuée</span>
    </button>
</div>

```

## Confirmation.php :

```
1 <?php
2 $dsn = 'mysql:host=localhost;dbname=site_e-commerce';
3 $username = 'root';
4 $password = '';
5
6 require_once "includes/config.php";
7 require_once "_header.php";
8
9
10 try {
11     $db = new PDO($dsn, $username, $password);
12 } catch (PDOException $e) {
13     echo 'Connexion échouée : ' . $e->getMessage();
14     exit;
15 }
16
17 $client_id = $_SESSION['id'];
18
19 // Mise à jour de l'état de la commande précédente
20 $update_commande = $db->prepare('UPDATE commande SET etat_commande = 1 WHERE client_id = :client_id AND etat_commande = 0');
21 $update_commande->execute(array('client_id' => $client_id));
22
23 // Crédation d'une nouvelle commande pour le client actuel
24 $date_commande = date('Y-m-d H:i:s');
25 $insert_commande = $db->prepare('INSERT INTO commande (date_commande, montant_commande, etat_commande, client_id) VALUES (:date_commande, 0, 0, :client_id)');
26 $insert_commande->execute(array('date_commande' => $date_commande, 'client_id' => $client_id));
27 $num_commande = $db->lastInsertId();
28 ?>
```

Ce code est une page de confirmation de commande d'un site e-commerce. Voici un résumé de ce qu'il fait :

1. Établit une connexion à la base de données MySQL en utilisant les informations de connexion fournies.
2. Récupère l'ID du client à partir de la variable de session.
3. Met à jour l'état de la commande précédente du client pour le marquer comme confirmée.
4. Crée une nouvelle commande pour le client actuel en insérant la date de commande et d'autres informations dans la table de commandes.
5. Récupère le numéro de la commande nouvellement créée à partir de la base de données.
6. Affiche une page HTML avec un message de confirmation de commande, comprenant une animation de coche de validation et un bouton pour retourner à la page d'accueil.

En résumé, ce code établit une connexion à la base de données, met à jour l'état de la commande précédente du client, crée une nouvelle commande et affiche une page de confirmation de commande.

## Dashboard.php :

Ce code est une page de profil utilisateur dans un site web de commerce électronique.  
Voici un résumé de ce que fait le code :

1. Il vérifie si l'utilisateur est connecté. Sinon, il redirige vers la page de connexion.
2. Il récupère les informations de l'utilisateur connecté à partir de la base de données.
3. Il récupère les commandes passées par l'utilisateur.
4. Il affiche les informations de profil de l'utilisateur, y compris le nom d'utilisateur et l'adresse e-mail.
5. Pour chaque commande de l'utilisateur avec un état de commande égal à 1, il affiche la date de la commande et le montant.
6. Pour chaque commande, il récupère les articles associés à cette commande et affiche les détails de chaque article, y compris l'image, le nom, la quantité et le prix.
7. Il affiche un lien de déconnexion pour permettre à l'utilisateur de se déconnecter du profil.



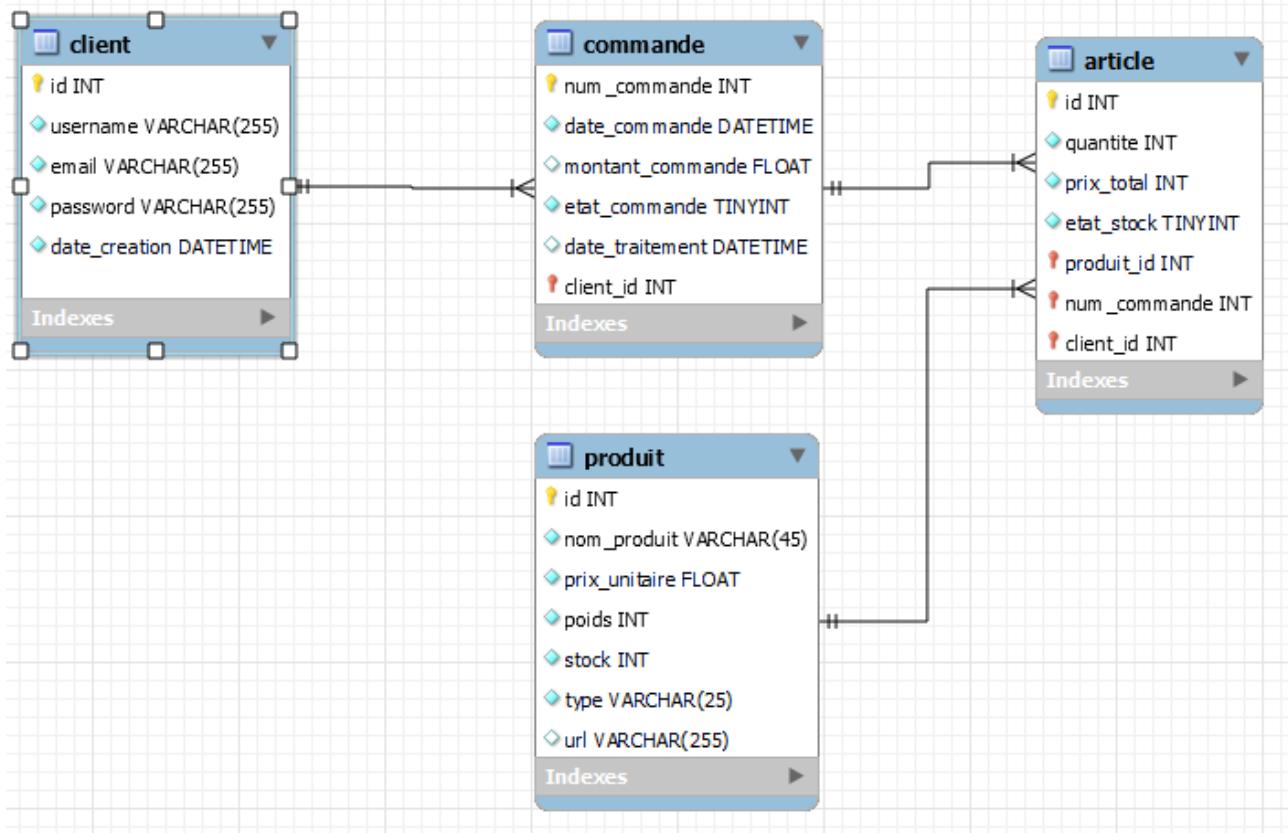
```
1 <?php
2 require_once('includes/config.php');
3
4 // Vérifier si l'utilisateur est connecté, sinon rediriger vers la page de connexion
5 if(!$userLoggedIn) {
6     header('Location: login.php');
7     exit;
8 }
9
10 // Récupérer les informations de l'utilisateur connecté depuis la base de données
11 $stmt = $db->prepare("SELECT username, email FROM client WHERE id = ?");
12 $stmt->bind_param("i", $userLoggedIn);
13 $stmt->execute();
14 $result = $stmt->get_result();
15 if($result->num_rows === 0) exit('Aucun utilisateur trouvé');
16 $user = $result->fetch_assoc();
17
18 // Récupérer les commandes passées par l'utilisateur
19 $stmt = $db->prepare("SELECT * FROM commande WHERE client_id = ?");
20 $stmt->bind_param("i", $userLoggedIn);
21 $stmt->execute();
22 $result = $stmt->get_result();
23 ?>
```

```

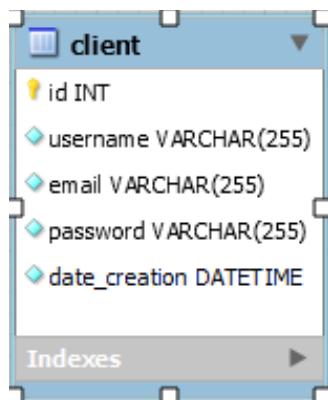
1 <h1 class="titre-profil">Bienvenue sur votre profil, <?= htmlspecialchars($user['username']) ?></h1>
2 <p class="adresse-email">Votre adresse e-mail : <?= htmlspecialchars($user['email']) ?></p>
3
4 <div class="commandes">
5   <?php
6   // Récupérer les commandes avec un état de commande égal à 1
7   $stmt = $db->prepare("SELECT * FROM commande WHERE client_id = ? AND etat_commande = 1");
8   $stmt->bind_param("i", $userLoggedIn);
9   $stmt->execute();
10  $result = $stmt->get_result();
11  while($row = $result->fetch_assoc()): ?>
12    <div class="commande">
13      <h2 class="commande-titre">Commande du <?= $row['date_commande'] ?></h2>
14      <p class="commande-montant">Montant : <?= $row['montant_commande'] ?> €</p>
15    </?php
16    // Récupérer les articles pour cette commande
17    $stmt = $db->prepare("SELECT * FROM article WHERE num_commande = ? AND client_id = ?");
18    $stmt->bind_param("ii", $row['num_commande'], $userLoggedIn);
19    $stmt->execute();
20    $articleResult = $stmt->get_result();
21  ?>
22
23 <ul class='articles'>
24   <?php while($articleRow = $articleResult->fetch_assoc()): ?>
25     <?php
26       // Récupérer les informations sur le produit associé à cet article
27       $stmt = $db->prepare("SELECT * FROM produit WHERE id = ?");
28       $stmt->bind_param("i", $articleRow['produit_id']);
29       $stmt->execute();
30       $produitResult = $stmt->get_result();
31       $produitRow = $produitResult->fetch_assoc();
32     ?>
33     <li class='article'>
34       ">
35       <div>
36         <p><?= $produitRow['nom_produit'] ?> (<?= $articleRow['quantite'] ?> <?= $produitRow['unite'] ?>)</p>
37         <p><?= $articleRow['prix_total'] ?> €</p>
38       </div>
39     </li>
40   <?php endwhile; ?>

```

## Base de données :



## Table Client :



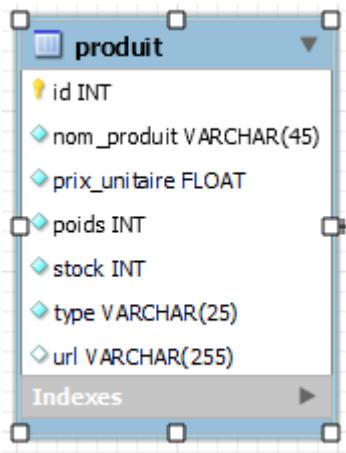
Ce code crée une table appelée "client" dans une base de données "site\_e-commerce". La table "client" a les colonnes suivantes : "id" (un entier auto-incrémenté), "username" (une chaîne de caractères non vide), "email" (une chaîne de caractères non vide), "password" (une chaîne de caractères non vide) et "date\_creation" (une date et heure non vide). La clé primaire de la table est l'ID. Le moteur de stockage utilisé est InnoDB. L'auto-incrémentation est activée avec une valeur de départ de 1. Le jeu de caractères par défaut utilisé est "utf8mb3".

## Table Produit :

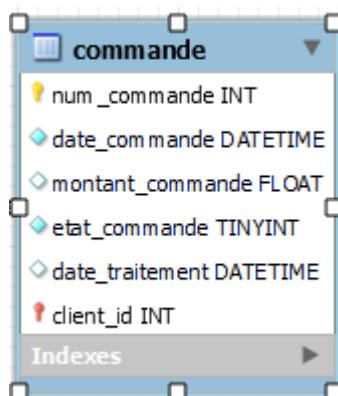
Ce code crée une table appelée "produit" dans une base de données nommée "site\_e-commerce". La table "produit" a les colonnes suivantes :

- `id` : un identifiant unique pour chaque produit (auto-incrémenté).
- `nom_produit` : le nom du produit (chaîne de caractères).
- `prix_unitaire` : le prix unitaire du produit (nombre décimal).
- `poids` : le poids du produit (nombre entier).
- `stock` : la quantité en stock du produit (nombre entier).
- `type` : le type du produit (chaîne de caractères).
- `url` : l'URL de l'image du produit (chaîne de caractères, peut être nulle par défaut).

La table est créée avec le moteur de stockage InnoDB et utilise l'encodage de caractères par défaut utf8mb3. L'attribut `AUTO_INCREMENT` est défini sur 1, ce qui signifie que la première valeur d'ID sera 1.



## Table Commande :



Ce code crée une table nommée "commande" dans une base de données appelée "site\_e-commerce". La table "commande" contient les colonnes suivantes :

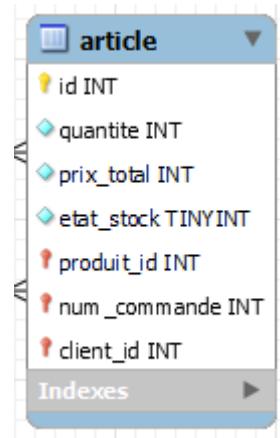
- `num_commande` : Un identifiant unique pour chaque commande, qui s'incrémentera automatiquement.
- `date_commande` : La date et l'heure auxquelles la commande a été passée.
- `montant_commande` : Le montant total de la commande, sous forme de nombre décimal.
- `etat_commande` : Un indicateur de l'état de la commande, représenté par un nombre entier (0 ou 1, par exemple).
- `date_traitement` : La date et l'heure auxquelles la commande a été traitée.
- `client_id` : L'identifiant du client associé à la commande.

La clé primaire de cette table est composée des colonnes `num_commande` et `client_id`, ce qui signifie que chaque combinaison de ces deux valeurs doit être unique. De plus, il y a une indexation sur la colonne `client_id` pour des performances optimales.

Il y a également une contrainte de clé étrangère nommée `fk_commande_client` qui lie la colonne `client_id` de la table "commande" à la colonne `id` de la table "client" de la même base de données. Cela signifie que la valeur de `client_id` dans la table "commande" doit correspondre à une valeur existante dans la table "client". De plus, aucune action de suppression ou de mise à jour n'est effectuée automatiquement sur la table "commande" lorsque des modifications sont apportées à la table "client".

Le moteur de stockage utilisé pour cette table est InnoDB, et la valeur de l'auto-incrémentation est fixée à 1, ce qui signifie que le premier enregistrement aura l'identifiant 1 et chaque nouvel enregistrement aura un identifiant incrémenté de 1.

### **Table Article :**



Ce code crée une table nommée "article" dans une base de données appelée "site\_e-commerce". La table "article" contient les colonnes suivantes :

- "id" : Un identifiant unique pour chaque article (auto-incrémenté).
- "quantite" : La quantité d'articles.
- "prix\_total" : Le prix total de l'article.
- "etat\_stock" : L'état du stock de l'article.
- "produit\_id" : L'identifiant du produit associé à l'article.
- "num\_commande" : Le numéro de commande associé à l'article.
- "client\_id" : L'identifiant du client associé à l'article.

La clé primaire de la table est composée des colonnes "id", "produit\_id", "num\_commande" et "client\_id". Des index sont également créés pour les colonnes "num\_commande", "client\_id" et "produit\_id" afin d'améliorer les performances des requêtes.

Des contraintes (foreign keys) sont définies pour garantir l'intégrité référentielle :

- La contrainte "fk\_article\_commande1" relie les colonnes "num\_commande" et "client\_id" de la table "article" aux colonnes correspondantes de la table "commande" dans la base de données "site\_e-commerce".
- La contrainte "fk\_article\_produit1" relie la colonne "produit\_id" de la table "article" à la colonne "id" de la table "produit" dans la base de données "site\_e-commerce".

Le moteur de stockage InnoDB est utilisé pour cette table. L'auto-incrémentation commence à partir de 1 et le jeu de caractères par défaut est "utf8mb3".

## Add.php :

```

1 $users_id = $_SESSION['id'];
2 $order_query = $pdo->prepare("SELECT * FROM commande WHERE client_id = ? AND etat_commande = 0");
3 $order_query->execute([$users_id]);
4 $order = $order_query->fetch(PDO::FETCH_ASSOC);
5
6 if (!$order) {
7     // Si l'utilisateur n'a pas de commande en cours, en créer une
8     $now = date('Y-m-d H:i:s');
9     $order_query = $pdo->prepare("INSERT INTO commande (date_commande, montant_commande, etat_commande, client_id) VALUES (?, ?, ?, ?)");
10    $order_query->execute([$now, 0, 1, $users_id]);
11
12    $order_id = $pdo->lastInsertId();
13 } else {
14     $order_id = $order['num_commande'];
15 }
16
17 // Vérifie si le produit est déjà dans le panier
18 $cart_item_query = $pdo->prepare("SELECT * FROM article WHERE produit_id = ? AND num_commande = ? AND client_id = ?");
19 $cart_item_query->execute([$product_id, $order_id, $users_id]);
20 $cart_item = $cart_item_query->fetch(PDO::FETCH_ASSOC);
21
22 if ($cart_item) {
23     // Si le produit est déjà dans le panier, mettre à jour la quantité
24     $new_quantity = $cart_item['quantite'] + $quantity;
25     $total_price = $new_quantity * $cart_item['prix_unitaire'];
26
27     $cart_item_update_query = $pdo->prepare("UPDATE article SET quantite = ?, prix_total = ? WHERE id = ?");
28     $cart_item_update_query->execute([$new_quantity, $total_price, $cart_item['id']]);
29 } else {
30     // Sinon, ajouter le produit au panier
31     $product_query = $pdo->prepare("SELECT * FROM produit WHERE id = ?");
32     $product_query->execute([$product_id]);
33     $product = $product_query->fetch(PDO::FETCH_ASSOC);
34
35     $total_price = $quantity * $product['prix_unitaire'];
36
37     $cart_item_insert_query = $pdo->prepare("INSERT INTO article (quantite, prix_total, produit_id, num_commande, client_id, etat_valable) VALUES (?, ?, ?, ?, ?, 1)");
38     $cart_item_insert_query->execute([$quantity, $total_price, $product_id, $order_id, $users_id]);
39 }
```

Établit une connexion à une base de données MySQL en utilisant les informations d'identification fournies (hôte, nom de la base de données, nom d'utilisateur et mot de passe).

1. Vérifie si l'utilisateur est connecté en vérifiant la présence d'une variable de session contenant le nom d'utilisateur. Si l'utilisateur n'est pas connecté, il est redirigé vers la page de connexion.
2. Si la requête HTTP est de type POST, le code récupère l'identifiant et la quantité d'un produit à partir des données soumises.
3. Vérifie si l'utilisateur a déjà une commande en cours en effectuant une requête SQL pour récupérer les informations de commande correspondantes.
4. Si l'utilisateur n'a pas de commande en cours, une nouvelle commande est créée avec la date actuelle, un montant initial de 0 et un état de commande de 1 (en cours). L'identifiant de la commande est récupéré.
5. Vérifie si le produit est déjà dans le panier de l'utilisateur en effectuant une requête SQL avec l'identifiant du produit, l'identifiant de la commande et l'identifiant de l'utilisateur.
6. Si le produit est déjà présent dans le panier, la quantité est mise à jour en ajoutant la quantité soumise à la quantité existante, et le prix total est recalculé en fonction de la nouvelle quantité. Les données du produit dans le panier sont mises à jour dans la base de données.
7. Si le produit n'est pas encore dans le panier, les informations du produit sont récupérées à partir de la base de données, et le prix total est calculé en multipliant la quantité soumise par le prix unitaire du produit. Les données du produit sont insérées dans la table des articles du panier.
8. Après avoir effectué les actions appropriées, l'utilisateur est redirigé vers la page du panier.
9. Si la méthode HTTP n'est pas POST, l'utilisateur est redirigé vers la page d'accueil.

## **F - Conclusion :**

Le site e-commerce "MarketMate" permet aux utilisateurs de parcourir une liste de produits, d'ajouter des produits au panier et de passer des commandes. Le site comporte plusieurs fonctionnalités telles que l'inscription et la connexion des utilisateurs.

Le code utilise PHP et MySQL pour la gestion des données. Il y a une base de données contenant des tables pour les produits, les utilisateurs, les commandes, etc. Le code effectue des requêtes SQL pour récupérer les données nécessaires à l'affichage des produits et à la gestion des utilisateurs.

Le site dispose d'un système d'authentification qui vérifie si l'utilisateur est connecté. Si un utilisateur n'est pas connecté, il est redirigé vers la page de connexion. Une fois connecté, l'utilisateur peut ajouter des produits au panier en spécifiant la quantité souhaitée.

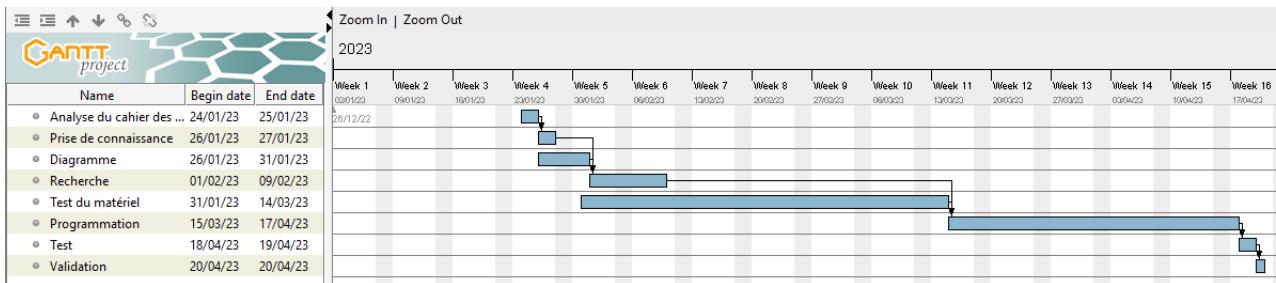
Le code utilise des boucles et des conditions pour afficher les produits, leurs images, leurs prix et les formulaires d'ajout au panier. Il y a également des fonctionnalités de mise à jour dynamique du prix total en fonction de la quantité sélectionnée.

Le site comprend également des pages d'inscription et de connexion pour les utilisateurs. Lors de l'inscription, les informations de l'utilisateur sont vérifiées et stockées dans la base de données. Lors de la connexion, les informations sont vérifiées et une session est démarrée pour maintenir l'état de connexion de l'utilisateur.

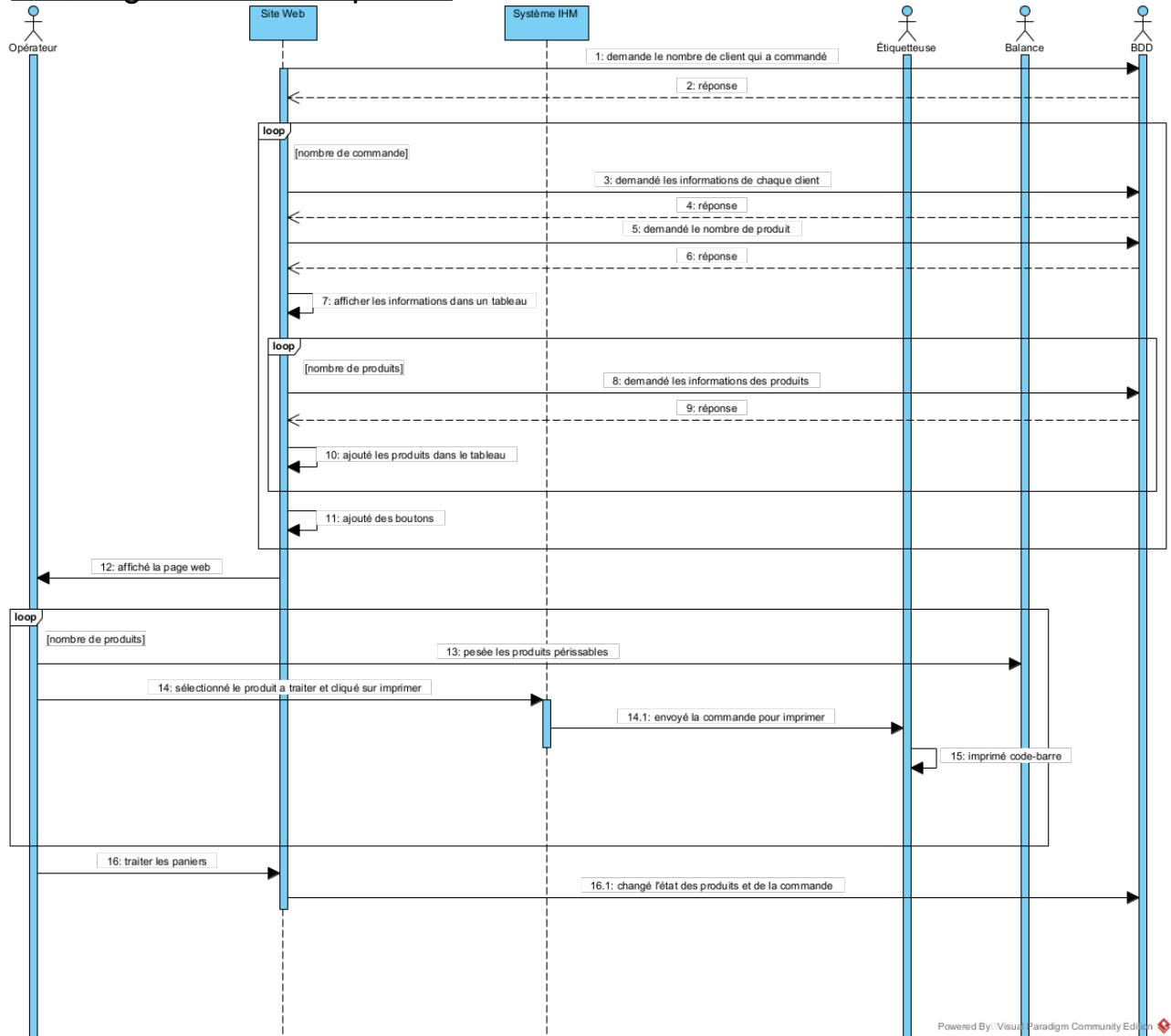
En conclusion, le code fourni représente une partie d'un site e-commerce fonctionnel. Cependant, il manque des éléments tels que la gestion des commandes, le processus de paiement et d'autres fonctionnalités importantes pour un site e-commerce complet. Il est également important de prendre en compte la sécurité du site en ajoutant des mesures de protection contre les attaques telles que l'injection SQL et la validation des données utilisateur.

## II - Étudiant 2: PERERA Githendra

### A – Planification



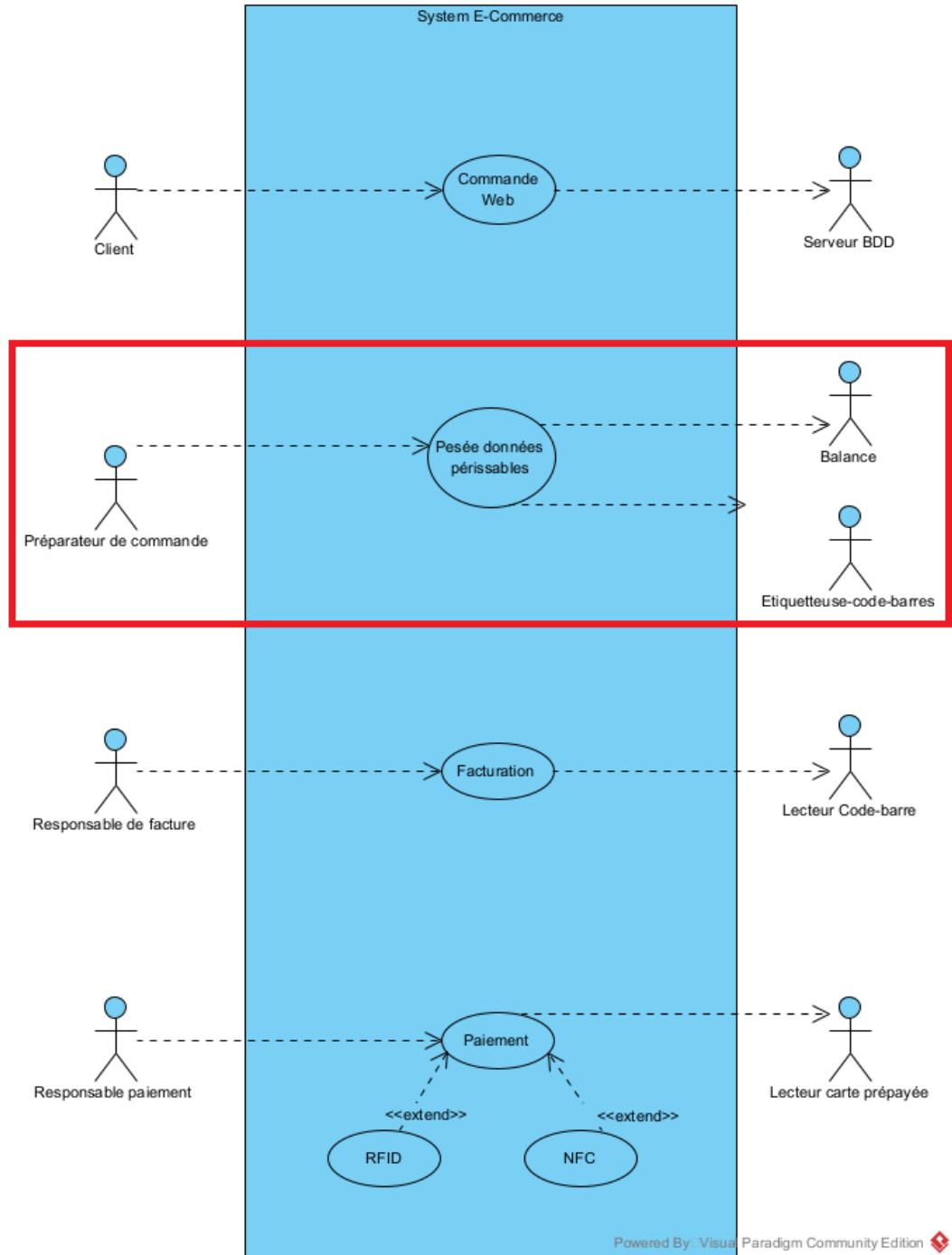
## B – Diagramme de séquence



Powered By: Visual Paradigm Community Edition

Ce diagramme de séquence représente le fonctionnement général de ma partie. Le but est traiter les paniers commandés des clients.

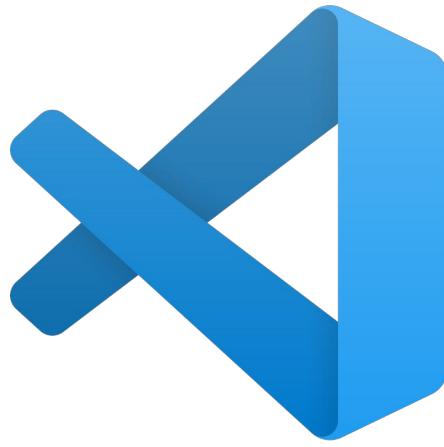
## C – Diagramme de cas d'utilisation



Ce diagramme de cas d'utilisation représente le fonctionnement du système en général.

Ma partie est celle des composants dans l'encadré rouge. C'est là que les équipements interviennent pour traiter les commandes des clients.

## D – Logiciels utilisés



Visual Studio Code, est un éditeur de code source développé par Microsoft. C'est un logiciel gratuit et open source, largement utilisé par les développeurs pour écrire, modifier et déboguer du code.



WampServer est un logiciel gratuit qui permet de mettre en place un environnement de développement web sur un ordinateur local. Il fournit une plateforme intégrée comprenant le serveur web Apache, le système de gestion de bases de données MySQL/MariaDB, ainsi que le langage de script PHP.



MySQL Workbench est un logiciel de gestion et d'administration de bases de données MySQL. Via une interface graphique intuitive, il permet de créer, modifier ou supprimer des tables, des comptes utilisateurs, et d'effectuer toutes les opérations inhérentes à la gestion d'une base de données. Pour ce faire, il doit être connecté à un serveur MySQL.

## E – Matériel principal de ma partie

Imprimante Zebra GK420d



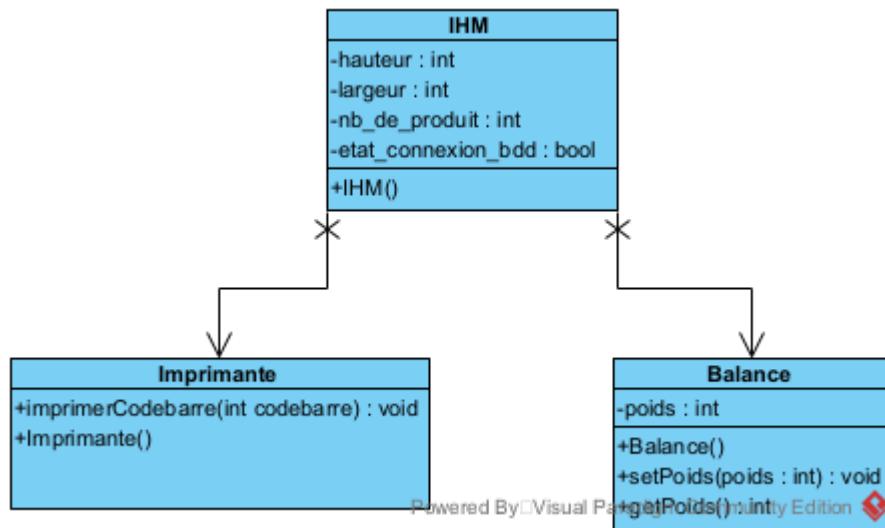
- Technologie d'impression thermique directe
- Résolution d'impression de 203 dpi (points par pouce)
- Interface USB et RS-232
- Taille de l'impression des étiquettes : 4,25 pouces
- Capacité du rouleau d'étiquettes

## Balance KERN 440-53N



- Capacité maximale de pesage de 50 kg
- Précision de pesée de 1 gramme
- Affichage numérique LCD rétroéclairé
- Interface RS-232

## F – Diagramme de classe de l'IHM



La classe IHM est composée de deux classes Imprimante et Balance, ce qui lui permet d'accéder à leurs méthodes pour communiquer avec ces équipements.

## G – Programme IHM

Classe Imprimante :

```
● ● ●  
1 from zebra import Zebra
```

La première ligne importe le module « Zebra » qui est utilisé pour communiquer avec l'imprimante.

```
● ● ●  
1 def __init__(self):  
2     self.z = Zebra('ZDesigner GK420d')
```

Cette méthode est le constructeur de la classe. Elle est appelée lorsqu'une nouvelle instance de la classe est créée. À l'intérieur du constructeur, une instance de la Zebra est créée avec le nom de l'imprimante utilisé comme argument, cette instance est utilisée pour communiquer avec l'imprimante.

```
1 def send_command(self, barcode):
2     if barcode:
3         # ZPL command
4         label = f"""
5             ^XA
6             ^FO170,50,^BY3
7             ^BEN,110,Y,N
8             ^FD{barcode}^FS
9             ^XZ
10            """
11
12     self.z.output(label)
```

La méthode `send_command(self, barcode)` est utilisée pour envoyer une commande d'impression à l'imprimante. Elle prend un argument `barcode` qui représente le code-barres que vous souhaitez imprimer. Si le code-barres n'est pas vide (`if barcode:`), alors une commande ZPL (Zebra Programming Language) est créée dans une variable `label`. Cette commande ZPL contient les instructions nécessaires pour imprimer le code-barres sur l'étiquette.

Enfin, la méthode `output(label)` de l'objet Zebra est appelée avec la variable `label` en tant qu'argument pour envoyer la commande d'impression à l'imprimante Zebra.

Classe Balance :

```
● ● ●  
1 def __init__(self):  
2     self.ser = serial.Serial()  
3     self.ser.port = "COM1"  
4     self.ser.baudrate = 9600  
5     self.ser.bytesize = serial.EIGHTBITS  
6     self.ser.parity = serial.PARITY_NONE  
7     self.ser.stopbits = serial.STOPBITS_ONE  
8     self.ser.timeout = 1
```

Dans la classe, le constructeur sert à configurer le port série de l'ordinateur pour que l'IHM puisse communiquer avec la balance.

```
● ● ●  
1 def setWeight(self):  
2     self.ser.open()  
3     self.ser.write(b"s")  
4  
5     data = self.ser.readline().decode('ascii')  
6  
7     if data:  
8         if data.strip().find("M") != -1: # == False  
9             self.scaleOutput = data.strip().replace("M      ","")  
10            self.weightProduct = int(self.scaleOutput.replace(" g",""))  
11        else:  
12            self.scaleOutput = data.strip().replace("      ","")  
13            self.weightProduct = int(self.scaleOutput.strip().replace(" g",""))  
14        else:  
15            self.weightProduct = None  
16  
17     self.ser.close()
```

La méthode setweight permet de récupérer le poids affiché sur la balance en utilisant la connexion port série avec l'ordinateur. Le poids est ensuite enregistré dans un attribut pour une utilisation ultérieure.



```
1 def getWeight(self):
2     return self.weightProduct
3
4 def getScaleOutput(self):
5     return self.scaleOutput
```

La méthode `getWeight` permet de renvoyer le poids entier à l'IHM afin qu'il puisse être utilisé pour générer le code-barre.

Et la méthode `getScaleOutput` permet de renvoyer le poids sous forme de chaîne de caractères à l'IHM pour son affichage.

Classe IHM :



```
1 from tkinter import *
2 import customtkinter as ctk
```

Pour afficher la fenêtre, nous avons besoin du module `tkinter` pour l'affichage et du module `customtkinter` pour rendre le style de la fenêtre plus moderne.



```
1 appWidth, appHeight = 460, 215
```

Nous devons d'abord déclarer deux attributs : la hauteur (appHeight) et la largeur (appWidth) de la fenêtre. L'attribut hauteur changera en fonction du nombre de boutons pour chaque produit. Cette action sera effectuée ultérieurement dans le code, avant l'ouverture de la fenêtre.



```
1 printer = Printer()  
2 scale = Scale()
```

Ensuite, nous créons des instances des classes Imprimante et Balance, que nous assignons aux objets printer et scale respectivement. Cela nous permet d'accéder aux méthodes de ces classes pour communiquer avec l'imprimante et la balance.

```

1  try:
2      self.mydb = mysql.connector.connect(
3          host = "localhost", #"mysql-projet-e-commerce.alwaysdata.net",
4          user = "root", #"312817",
5          password = "", #LucasRatonLaveur",
6          database = "site_e-commerce" #"projet-e-commerce_bdd"
7      )
8      self.mycursor = self.mydb.cursor()
9      self.dbConnectionState = True
10
11 except mysql.connector.Error as e:
12     print("Error reading data from MySQL table", e)
13     self.dbConnectionState = False
14     self.overrideredirect(1)
15     CTkMessagebox(title="Erreur", message="Échec connexion BDD", icon="cancel")
16

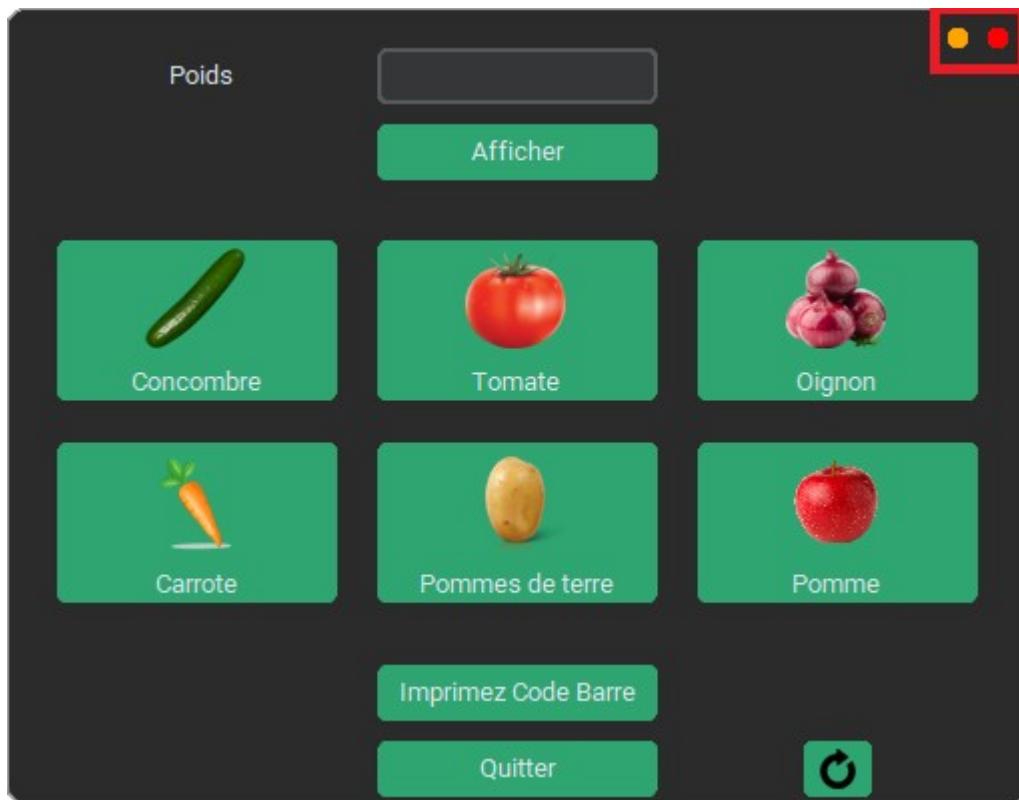
```

Pour communiquer avec la base de données, nous tentons d'établir une connexion. Si la connexion réussit, nous mettons à jour l'état de la variable dbConnectionState à True, ce qui signifie que la connexion est établie. Sinon, l'interface utilisateur affichera une fenêtre d'erreur indiquant qu'elle n'a pas pu se connecter à la base de données.

```

1  if self.dbConnectionState is True:
2      self.attributes("-topmost", True)
3
4      self.frame_top = ctk.CTkFrame(self, corner_radius = 10, width = 10, border_width=1)
5      self.frame_top.grid(sticky="nsew")
6
7      self.frame_top.bind("<B1-Motion>", self.move_window)
8      self.frame_top.bind("<ButtonPress-1>", self.oldxyset)
9      self.frame_top.bind("<Map>", self.on_configure)
10
11     self.button_close = ctk.CTkButton(self.frame_top, corner_radius=10, width=10, height=10, hover=False,
12                                         text="", fg_color="red", command=self.button_event)
13     self.button_close.configure(cursor="arrow")
14     self.button_close.grid(row=0, column=3, sticky="ne", padx=(5, 10), pady=10)
15     self.button_close.bind("<Enter>", lambda e, button = self.button_close: self.enter_closeButton(button))
16     self.button_close.bind("<Leave>", lambda e, button = self.button_close: self.leave_closeButton(button))
17
18     self.button_minimize = ctk.CTkButton(self.frame_top, corner_radius=10, width=10, height=10, hover=False,
19                                         text="", fg_color="orange", command=self.minimize_window)
20     self.button_minimize.configure(cursor="arrow")
21     self.button_minimize.grid(row=0, column=2, sticky="ne", padx=5, pady=10)
22

```



Après avoir essayé de se connecter à la base de données, nous vérifions ensuite si la connexion a été établie avec succès en vérifiant l'état de la variable. Si la variable est vraie, alors nous pouvons continuer. Nous allons d'abord ajouter un bouton pour fermer l'interface utilisateur et un bouton pour réduire la fenêtre. Ensuite, nous allons configurer ces boutons pour appeler une fonction lorsque l'utilisateur appuie dessus.

```

● ○ ●

1 def button_event(self, event=None):
2     if self.fade:
3         self.fade_out()
4     self.grab_release()
5     self.mydb.close()
6     self.destroy()
7     self.event = event

```

Quand le bouton rouge est appuyé, il appelle la fonction 'button\_event'. Cette fonction ferme d'abord la connexion avec la base de données avant de fermer la fenêtre, afin de maintenir l'intégrité des données et d'assurer une fin appropriée du programme

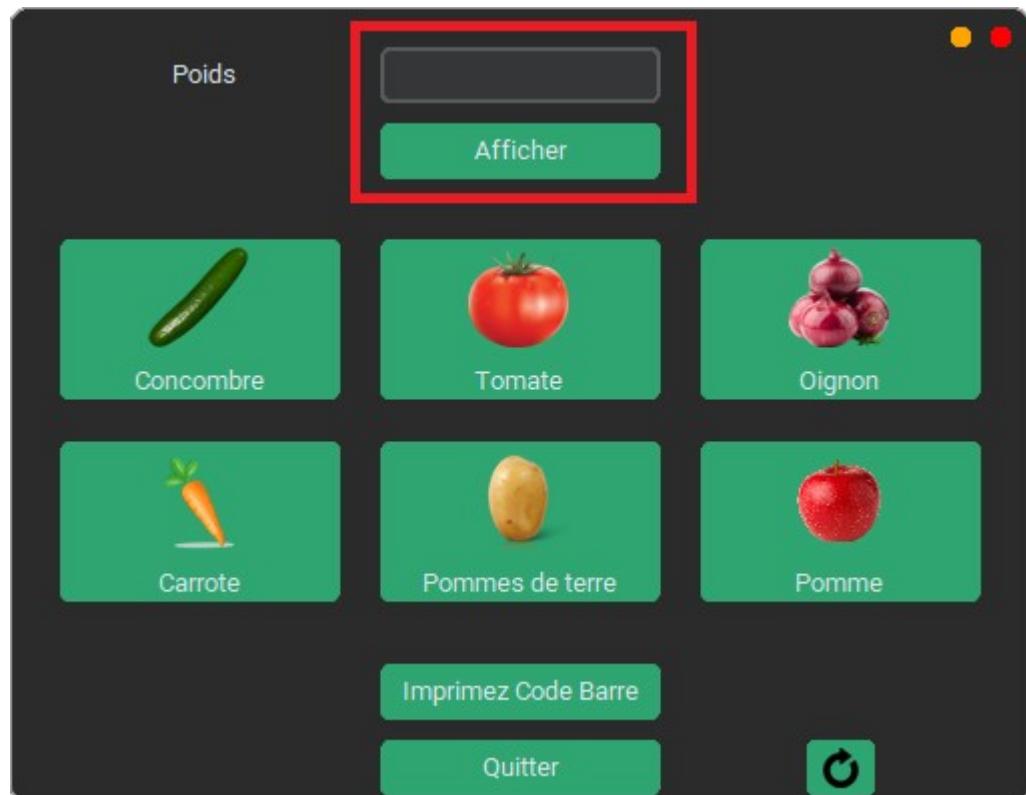
```
1 def minimize_window(self):
2     self.overrideredirect(0)
3     self.iconify()
```

Et quand le bouton jaune est appuyé, il appelle la fonction 'minimize\_window'. Cette fonction utilisée pour minimiser la fenêtre en l'icônisant, réduisant ainsi sa taille et en la plaçant dans la barre des tâches ou le dock du système d'exploitation.

```
1 self.mycursor.execute("SELECT MIN(id) AS first_id FROM produit")
2 firstID = int(str(self.mycursor.fetchone()).replace("(", "").replace(")", ""))
3
4 # get the id of the last perishable product from the produit table
5 self.mycursor.execute("SELECT MAX(id) AS first_id FROM produit;")
6 lastID = int(str(self.mycursor.fetchone()).replace("(", "").replace(")", ""))
7
8 for i in range(firstID, lastID + 1):
9     # get the name of the products stored in the database and add it in an array
10    self.mycursor.execute(f"SELECT nom_produit FROM produit WHERE id = {i} AND TYPE = 'périsable'")
11    nameItem = (str(self.mycursor.fetchone())).replace("(", "").replace(")", "")
12    if nameItem != "None":
13        self.productsArr[0].append(nameItem)
14        self.numberOfProducts += 1
15
16    # get the url of the picture stored in the database and add it in an array
17    self.mycursor.execute(f"SELECT URL FROM produit WHERE id = {i} AND TYPE = 'périsable'")
18    url = (str(self.mycursor.fetchone())).replace("(", "").replace(")", "")
19    if url != "None":
20        self.productsArr[1].append(url)
21
```

Nous allons récupérer tous les identifiants des produits en utilisant des commandes SQL, puis utiliser ces identifiants pour récupérer le nom et l'URL des produits. Ces informations seront stockées dans une table.

```
1 # Weight Label
2 weightLabel = ctk.CTkLabel(self.frame_top,
3                             text = "Poids")
4 weightLabel.grid(row = 0, column = 0,
5                   padx = (50, 20), pady = (20, 10),
6                   sticky = "ew")
7
8 # Weight Entry Field
9 self.weightEntry = ctk.CTkEntry(self.frame_top,
10                                state = 'disabled')
11 self.weightEntry.grid(row = 0, column = 1,
12                         columnspan = 1, padx = (20, 20),
13                         pady = (20, 10), sticky = "ew")
14
15 # Display the weight Button
16 weightButton = ctk.CTkButton(self.frame_top, text = "Afficher",
17                               command = self.insertWeight)
18 weightButton.grid(row = 1, column = 1,
19                     columnspan = 1,
20                     padx = (20, 20), pady = (0, 20),
21                     sticky = "ew")
22
```



Nous allons ajouter une label pour afficher le poids des produits pesée avec un bouton qui servira à récupérer le poids afficher sur la balance.

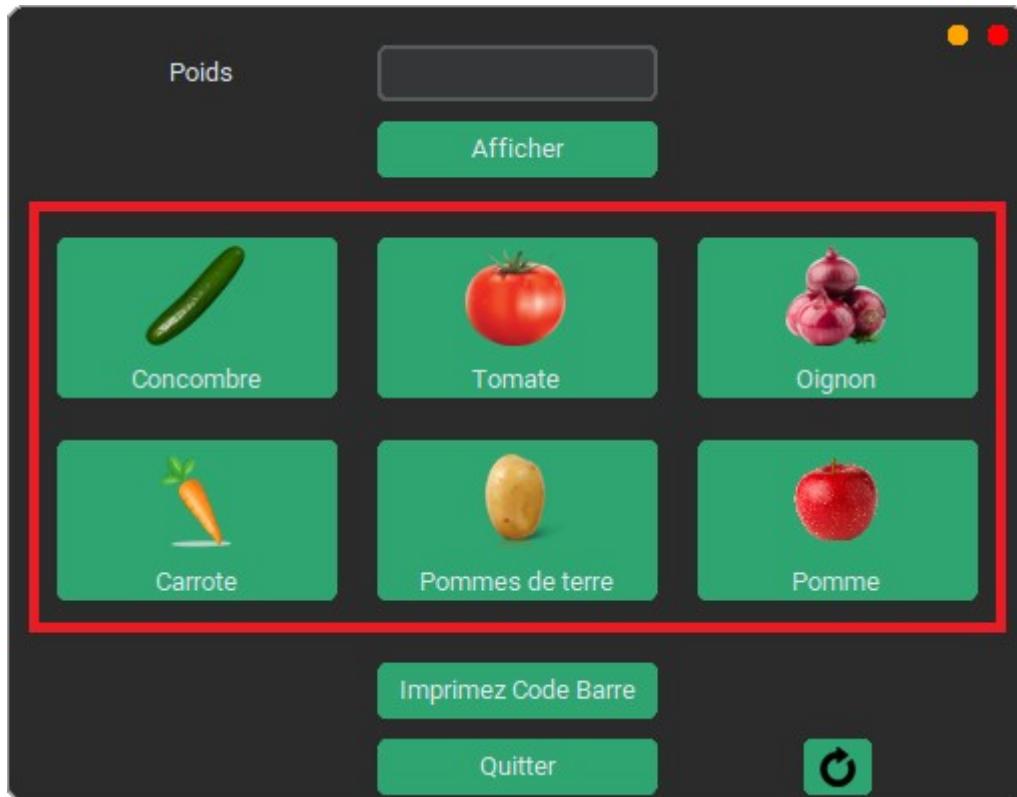
```
1 def insertWeight(self):
2     self.scale.setWeight()
3     text = self.scale.getScaleOutput()
4
5     if text:
6         self.weightEntry.configure(state = 'normal', justify = CENTER)
7         self.weightEntry.delete(0, "end")
8         self.weightEntry.insert(1, text)
9         self.weightEntry.configure(state = 'disabled')
10    else:
11        self.weightEntry.configure(state = 'normal', justify = CENTER)
12        self.weightEntry.delete(0, "end")
13        self.weightEntry.insert(1, "Erreur")
14        self.weightEntry.configure(state = 'disabled')
15
```

Pour afficher le poids, le bouton "Afficher" appelle une fonction qui récupère le poids de la balance en utilisant les méthodes de la classe de la balance. Ce poids est ensuite stocké dans une variable appelée 'text'. Ensuite, l'interface utilisateur vérifie si cette variable est vide ou non. Si elle est vide, cela signifie que le poids n'a pas été récupéré avec succès, et la label affiche un message d'erreur. Sinon, si la variable contient une valeur, la label affiche le contenu de cette variable, qui représente le poids du produit.

```

1      for i in range(self.numberOfProducts):
2
3          # buttons and resize images
4          img_url = self.productsArr[1][i]
5          img = self.load_image_from_url(img_url, IMG_SIZE)
6
7          self.button = ctk.CTkButton(self.frame_top,
8              image = img, text = self.productsArr[0][i],
9              compound = "top", command = lambda productName = self.productsArr[0][i]: self.get_product_id(productName))
10
11         # detect the mouse hovering the buttons
12         self.button.bind("<ButtonPress-1>", lambda e, buttons = self.buttonsArr[0], button = self.button: self.on_click(button, buttons))
13         self.button.bind("<Enter>", lambda e, buttons = self.buttonsArr[0], button = self.button: self.buttonEnterHover(button, buttons))
14         self.button.bind("<Leave>", lambda e, buttons = self.buttonsArr[0], button = self.button: self.buttonLeaveHover(button, buttons))
15         self.button.bind("<ButtonRelease-1>", lambda e, buttons = self.buttonsArr[0], button = self.button: self.buttonClicked(button, buttons))
16
17         if i == 0:
18             self.appHeight += 100
19         else:
20             if i % BUTTONS_IN_A_ROW == 0:
21                 self.rowX += 1
22                 self.columnY = 0
23                 self.appHeight += 100
24
25         if i == 0:
26             self.button.grid(column = self.columnY, row = self.rowX,
27                               columnspan = 1, padx = (25, 0), pady = 10)
28         else:
29             self.button.grid(column = self.columnY, row = self.rowX,
30                               columnspan = 1, padx = (25 if (i % BUTTONS_IN_A_ROW == 0) else 0, 0), pady = 10)
31
32         self.buttonsArr[0].append(self.button)
33         self.buttonsArr[1].append("green")
34         self.columnY += 1
35

```

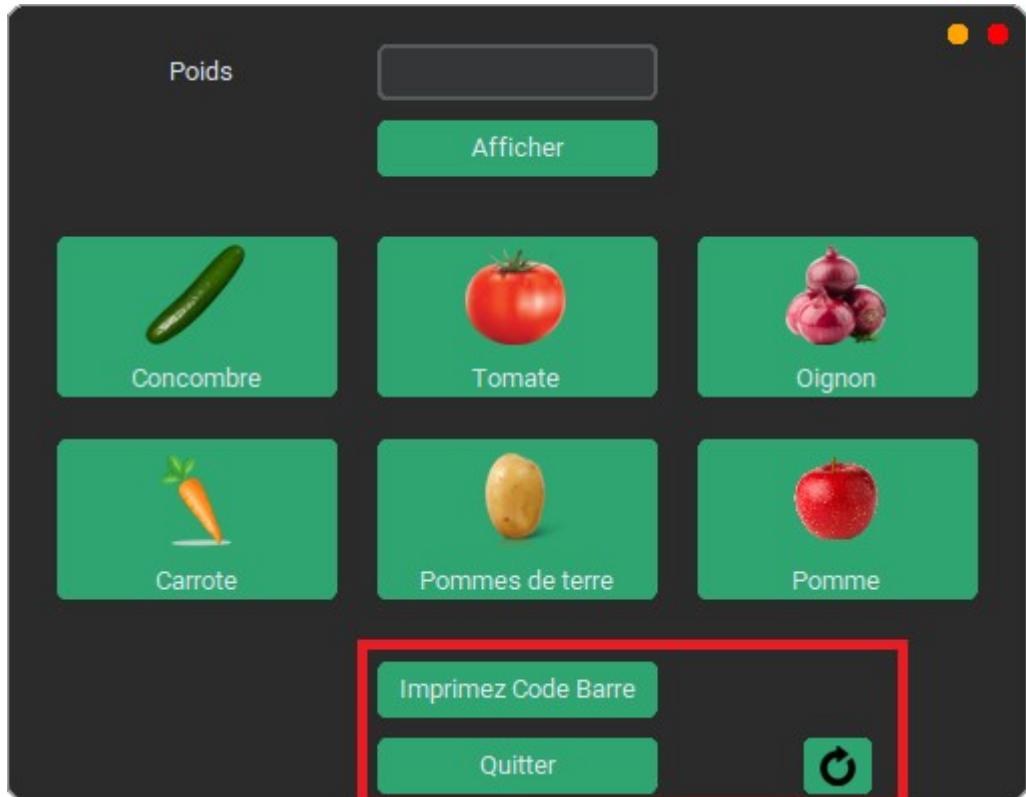


En dessous du bouton, nous allons ajouter des boutons pour chaque produit enregistré dans la base de données. Nous utiliserons les données stockées dans la table pour afficher le nom et l'image du produit. Ces boutons vont enregistrer l'identifiant du produit dans un attribut dès que l'utilisateur choisit le produit pesé.

```
1 def load_image_from_url(self, url, new_size):
2     response = requests.get(url)
3     image = Image.open(BytesIO(response.content))
4     image = image.resize(new_size)
5     return ImageTk.PhotoImage(image)
```

Pour afficher les images, l'IHM appellera la fonction ‘load\_image\_from\_url’ qui utilise le module ‘requests’. Cette fonction est utilisée pour récupérer le contenu de l'image à partir de l'URL spécifiée. Ensuite, elle utilise la bibliothèque PIL (Pillow) pour ouvrir l'image à partir du contenu récupéré. L'image est ensuite redimensionnée pour s'adapter à la taille du bouton dans lequel elle sera affichée. Cette fonction est appelée plusieurs fois, une fois pour chaque produit, afin de charger et afficher les images correspondantes dans les boutons correspondants de l'interface.

```
1 printButton = ctk.CTkButton(self.frame_top,
2                             command = self.print_barcode,
3                             text = "Imprimez Code Barre")
4 printButton.grid(row = self.rowX + 1, column = 1,
5                   columnspan = 1,
6                   padx = (20, 20), pady = (20, 5),
7                   sticky = "ew")
8
9 quitButton = ctk.CTkButton(self.frame_top, text = "Quitter",
10                           command = self.button_event)
11 quitButton.grid(row = self.rowX + 2, column = 1,
12                  columnspan = 1,
13                  padx = (20, 20), pady = 5,
14                  sticky = "ew")
15
16 reloadButton = ctk.CTkButton(self.frame_top,
17                               image = self.load_image_from_url("https://cdn-icons-png.flaticon.com/512/560/560450.png", (20, 20)),
18                               text = None,
19                               width = 10, height = 10,
20                               command = self.reload)
21 reloadButton.grid(row = self.rowX + 2, column = 2,
22                   padx = (20, 20), pady = 5)
```



À la fin, nous ajouterons un bouton pour imprimer le code-barres. Ce bouton appellera la fonction 'print\_barcode' qui générera le code-barres et l'imprimera. Nous aurons également un bouton pour fermer la fenêtre qui utilise la même fonction que le petit bouton rouge et un bouton pour redémarrer l'interface utilisateur si la table des produits de la base de données a été modifiée.

```
● ● ●  
1 def reload(self):  
2     self.button_event()  
3     python = sys.executable  
4     subprocess.call([python, __file__])  
5     sys.exit()
```

Pour redémarrer l'IHM, le bouton appelle la fonction "reload". Cette fonction utilise la bibliothèque "subprocess" pour exécuter un nouveau processus Python en exécutant à nouveau le même script en cours (file). Cela a pour effet de relancer essentiellement le programme lui-même.

Ensuite, la fonction "sys.exit()" est appelée pour terminer l'exécution du programme actuel. Cela assure une sortie propre et la fermeture de tous les processus associés au programme.



```
1 def print_barcode(self):
2     self.printer.send_command(self.generate_barcode(self.idProduct, self.scale.getWeight()))
3
```

La fonction `print_barcode` lorsqu'elle est exécutée, elle utilise la classe `Printer` (Imprimante) pour envoyer une commande d'impression du code-barres. Cette commande est générée en appelant la fonction `generate_barcode` et `self.Scale.getWeight()`.

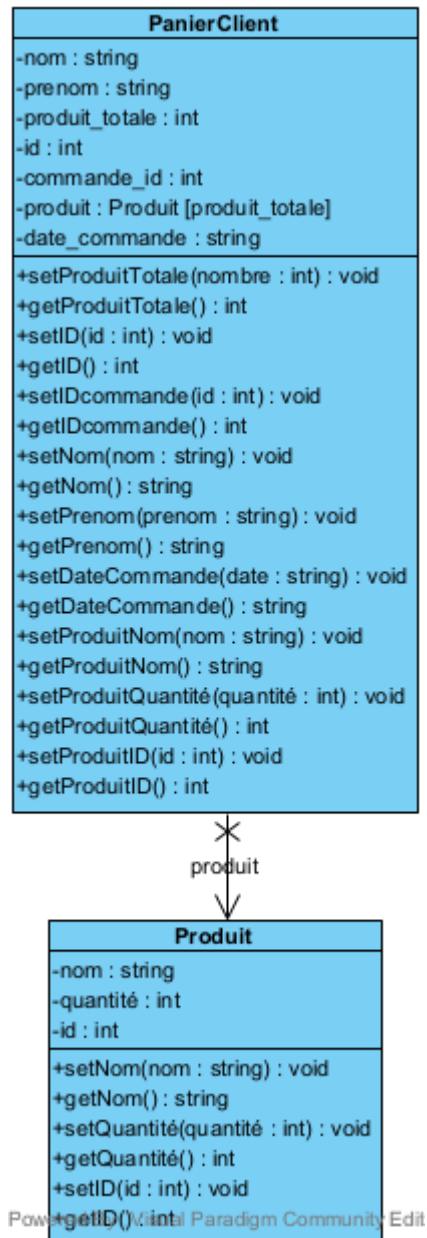


```
1 def generate_barcode(self, product_id, product_weight):
2     if product_id is None:
3         CTkMessageBox(message = "Choisissez un produit",
4                       icon = "warning", option_1 = "Ok")
5     return None
6
7     elif product_weight is None:
8         CTkMessageBox(message = "Récupérez le poids du produit",
9                       icon = "warning", option_1 = "Ok")
10    return None
11
12    barcode = int("{:06d}{:05d}".format(product_id, product_weight))
13    return barcode
```

Pour générer le code-barres, l'interface utilisateur appelle une fonction avec deux paramètres : l'identifiant du produit (choisi par l'utilisateur) et le poids du produit (récupéré en appuyant sur le bouton "Afficher").

La fonction vérifie d'abord si l'utilisateur a bien choisi un produit et a récupéré le poids. Si l'une de ces deux conditions n'est pas remplie, une fenêtre d'erreur s'affiche. Sinon, la fonction génère un code-barres en combinant l'identifiant du produit et le poids du produit. Elle formate ces valeurs en une chaîne spécifique avec six chiffres pour l'identifiant du produit et cinq chiffres pour le poids du produit.

## H – Diagramme de Classe pour l'affichage des commandes des clients

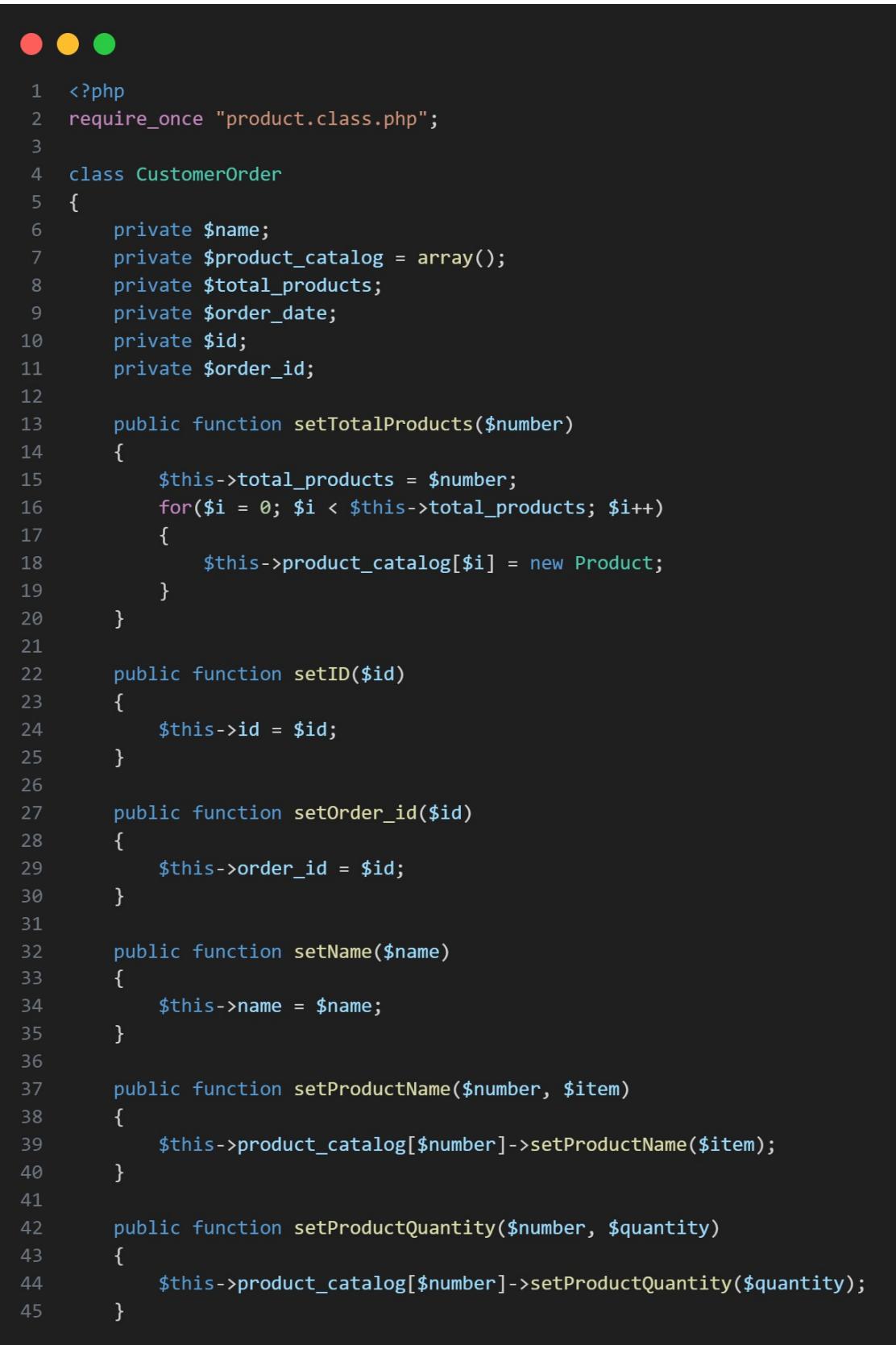


La classe PanierClient est composée de la classe Produit. Elle a pour rôle de stocker et utiliser les informations de chaque client ainsi que de ses commandes.

## J – Programme de l'affichage des commandes des clients

Pour afficher les commandes des clients, nous allons utiliser du PHP pour récupérer les commandes de chaque client.

Classe PanierClient :



```
 1 <?php
 2 require_once "product.class.php";
 3
 4 class CustomerOrder
 5 {
 6     private $name;
 7     private $product_catalog = array();
 8     private $total_products;
 9     private $order_date;
10    private $id;
11    private $order_id;
12
13    public function setTotalProducts($number)
14    {
15        $this->total_products = $number;
16        for($i = 0; $i < $this->total_products; $i++)
17        {
18            $this->product_catalog[$i] = new Product;
19        }
20    }
21
22    public function setID($id)
23    {
24        $this->id = $id;
25    }
26
27    public function setOrder_id($id)
28    {
29        $this->order_id = $id;
30    }
31
32    public function setName($name)
33    {
34        $this->name = $name;
35    }
36
37    public function setProductName($number, $item)
38    {
39        $this->product_catalog[$number]->setProductName($item);
40    }
41
42    public function setProductQuantity($number, $quantity)
43    {
44        $this->product_catalog[$number]->setProductQuantity($quantity);
45    }

```



```
1  public function setProductId($number, $product_id)
2  {
3      $this->product_catalog[$number]->setProductId($product_id);
4  }
5
6  public function setProductAvailable($number, $id)
7  {
8      $this->product_catalog[$number]->setProductAvailability($id);
9  }
10
11 public function setOrder_date($datetime)
12 {
13     $this->order_date = $datetime;
14 }
15
16 public function getID()
17 {
18     return $this->id;
19 }
20
21 public function getOrder_id()
22 {
23     return $this->order_id;
24 }
25
26 public function getName()
27 {
28     return $this->name;
29 }
30 public function getTotalProducts()
31 {
32     return $this->total_products;
33 }
34
35 public function getProductName($item_number)
36 {
37     return $this->product_catalog[$item_number]->getProductName();
38 }
39
40 public function getProductQuantity($item_number)
41 {
42     return $this->product_catalog[$item_number]->getProductQuantity();
43 }
44
45 public function getProductId($item_number)
46 {
47     return $this->product_catalog[$item_number]->getProductId();
48 }
49
50 public function getProductAvailable($item_number)
51 {
52     return $this->product_catalog[$item_number]->getProductAvailability();
53 }
54
55 public function getOrder_date()
56 {
57     return $this->order_date;
58 }
59 }
60 ?>
```

## Classe Produit :

```
1  <?php
2  class Product
3  {
4      private $name;
5      private $quantity;
6      private $id;
7      private $availability;
8
9      public function setProductName($name)
10     {
11         $this->name = $name;
12     }
13
14     public function setProductQuantity($quantity)
15     {
16         $this->quantity = $quantity;
17     }
18
19     public function setProductId($id)
20     {
21         $this->id = $id;
22     }
23
24     public function setProductAvailability($availability)
25     {
26         $this->availability = $availability;
27     }
28
29     public function getProductName()
30     {
31         return $this->name;
32     }
33
34     public function getProductQuantity()
35     {
36         return $this->quantity;
37     }
```

```
1     public function getProductId()
2     {
3         return $this->id;
4     }
5
6     public function getProductAvailability()
7     {
8         return $this->availability;
9     }
10    }
11    ?>
```

Pour récupérer les informations de chaque client et de ses commandes, nous allons tout d'abord nous connecter à la base de données.

```
1 $servername = "localhost";
2 $username = "root";
3 $password = "";
4 $dbname = "site_e-commerce";
5
6 $link = new mysqli($servername, $username, $password, $dbname);
```

Ensuite, il faut vérifier si la connexion a été bien établie. Si une erreur de connexion survient, le site affichera un message d'erreur. Sinon, le programme continue son exécution.

```
1 if ($link->connect_error)
2 {
3     error_reporting("Erreur avec les client_id" . $link->connect_error);
4 }
5 else
6 {
```

Une fois que le programme a établi une connexion à la base de données, il utilisera des commandes SQL pour récupérer les informations. Tout d'abord, il faudra récupérer le nombre de commandes à traiter afin d'utiliser une boucle qui servira à récupérer les identifiants de chaque commande. Ces identifiants seront stockés dans un tableau dans l'ordre du plus ancien au plus récent.

```
● ● ●  
1 $number_of_orders = $result->fetch_assoc()["COUNT(num_commande)"];  
2  
3 if ($number_of_orders > 0)  
4 {  
5     $order_id = array();  
6     $result = $link->query("SELECT num_commande FROM commande WHERE etat_commande = 1 ORDER BY date_commande ASC");  
7     while($row = $result->fetch_assoc())  
8     {  
9         $order_id[] = $row["num_commande"];  
10    }  
11  
12    $customerOrder = array();
```

Puis, le programme stocke les informations de chaque commande dans la classe PanierClient en utilisant ses méthodes.

```
● ● ●  
1 for ($i = 0; $i < $number_of_orders; $i++)  
2 {  
3     $customerOrder[$i] = new CustomerOrder();  
4     $customerOrder[$i]->setOrder_id($order_id[$i]);  
5  
6     $get_client_id = $link->query("SELECT client_id FROM commande WHERE num_commande = {$customerOrder[$i]->getOrder_id()}");  
7     $customerOrder[$i]->setID($get_client_id->fetch_assoc()["client_id"]);  
8  
9     $get_client_id = $link->query("SELECT client_id FROM commande WHERE num_commande = {$customerOrder[$i]->getOrder_id()}");  
10  
11    $get_date_order = $link->query("SELECT date_commande FROM commande WHERE num_commande = {$customerOrder[$i]->getOrder_id()}");  
12    $customerOrder[$i]->setOrder_date($get_date_order->fetch_assoc()["date_commande"]);  
13  
14    $get_username = $link->query("SELECT username FROM client WHERE id = '{$customerOrder[$i]->getOrder_id()}'");  
15    $customerOrder[$i]->setName($get_username->fetch_assoc()["username"]);  
16}
```

```
● ● ●  
1 $get_number_of_products = $link->query("SELECT COUNT(nom_produit) FROM produit WHERE id IN (SELECT produit_id FROM article WHERE num_commande = {$customerOrder[$i]->getOrder_id()})");  
2 $customerOrder[$i]->setTotalProducts($get_number_of_products->fetch_assoc()["COUNT(nom_produit)"]);  
3  
4 $get_products_name = $link->query("SELECT nom_produit FROM produit WHERE id IN (SELECT produit_id FROM article WHERE num_commande = {$customerOrder[$i]->getOrder_id()})");  
5  
6 $count = 0;  
7 while ($row = $get_products_name->fetch_assoc())  
8 {  
9     $customerOrder[$i]->setProductName($count, $row["nom_produit"]);  
10    $count++;  
11}  
12
```

```

1   for($j = 0; $j < $customerOrder[$i]->getTotalProducts(); $j++)
2   {
3       $get_product_quantity = $link->query("SELECT quantite FROM article WHERE num_commande = {$customerOrder[$i]->getOrder_id()} AND produit_id IN (SELECT id FROM produit WHERE nom_produit = '{$customerOrder[$i]->getProductName($j)}')");
4       $customerOrder[$i]->setProductQuantity($j, $get_product_quantity->fetch_assoc()["quantite"]);
5
6       $get_product_id = $link->query("SELECT id FROM article WHERE num_commande = {$customerOrder[$i]->getOrder_id()} AND produit_id IN (SELECT id FROM produit WHERE nom_produit = '{$customerOrder[$i]->getProductName($j)}')");
7       $customerOrder[$i]->setProductId($j, $get_product_id->fetch_assoc()["id"]);
8
9       $result = $link->query("SELECT IF(etat_stock = 1, 'true', 'false') AS non_manquant_checked FROM article WHERE id = {$customerOrder[$i]->getProductId($j)}");
10      $customerOrder[$i]->setProductAvailable($j, $result->fetch_assoc()["non_manquant_checked"]);
11
12  }
13

```

Pour afficher ces informations, le programme va créer une table pour chaque commande et y afficher les données correspondantes.

```

1  for ($i = 0; $i < $number_of_orders; $i++)
2  {
3      ?>
4      <table>
5          <tr>
6              <th colspan="3"><?php echo $customerOrder[$i]->getName(); ?></th>
7          </tr>
8          <tr>
9              <th>Produit</th>
10             <th>Quantité</th>
11             <th><?php echo $customerOrder[$i]->getOrder_date(); ?></th>
12         </tr>
13         <?php
14
15         for ($j = 0; $j < $customerOrder[$i]->getTotalProducts(); $j++)
16     {
17         $product_name = $customerOrder[$i]->getProductName($j);
18         $product_quantity = $customerOrder[$i]->getProductQuantity($j);
19         $product_id = $customerOrder[$i]->getProductId($j);
20         $order_id = $customerOrder[$i]->getOrder_id();
21         ?>
22         <tr>
23             <td><?php echo $product_name; ?></td>
24             <td><?php echo $product_quantity; ?></td>

```

Puis, pour permettre à l'utilisateur de sélectionner les produits valables, un bouton radio est ajouté pour chaque produit. Ainsi, l'utilisateur peut cocher les produits souhaités.

```

1  <td><input type="checkbox" name=<?php echo $product_id ;?>
2      class="checkbox"
3      data-order-id=<?php echo $order_id; ?>
4      data-product-quantity=<?php echo $product_quantity; ?>
5      id=<?php echo $checkbox_id ?> <?php if($customerOrder[$i]->getProductAvailable($j) == "true") { echo "checked"; } ?>></td>
6

```

Enfin, pour chaque commande, un bouton est ajouté afin de traiter les commandes et gérer les stocks. Ce bouton appelle une fonction JavaScript pour traiter la commande et mettre à jour les stocks des produits.

```
1 <table style="border-color: transparent;">
2     <tr>
3         <th colspan="3" style="background-color : transparent;">
4             <button id=<?php echo $customerOrder[$i]->getOrder_id(); ?>" onclick="updateNonManquant(<?php echo $customerOrder[$i]->getOrder_id(); ?, <?php echo $customerOrder[$i]->getTotalProducts();?>)">Valider</button>
5         </th>
6     </tr>
7 </table>
```

Cette fonction envoie des commandes SQL à un script PHP pour mettre à jour les stocks des produits et traiter les paniers. La fonction vérifie d'abord quel bouton est cliqué afin de récupérer le bon identifiant du panier, puis elle gère les stocks des produits valides et commandés par le client.

```
1 function updateNonManquant(order_id, totalProducts)
2 {
3     for (let i = 0; i < totalProducts; i++)
4     {
5         var checkbox = document.getElementById(i);
6         var product_id = checkbox.name;
7         var button = checkbox.getAttribute("data-order-id");
8         var product_quantity = checkbox.getAttribute("data-product-quantity");
9
10        if (button.includes(order_id))
11        {
12            var date = new Date().toISOString().slice(0, 19).replace('T', ' ');
13            var xhr = makeRequest();
14            var sql;
15
16            if (checkbox.checked)
17            {
18                sql = `UPDATE article SET etat_stock = 1 WHERE id = ${product_id} AND num_commande = ${order_id}`;
19                xhr.send(JSON.stringify({sql: sql}));
20
21                xhr = makeRequest();
22
23                sql = `UPDATE produit SET stock = stock - ${product_quantity} WHERE id = ${product_id}`;
24                xhr.send(JSON.stringify({sql: sql}));
25            }
26            else
27            {
28                sql = `UPDATE article SET etat_stock = 0 WHERE id = ${product_id} AND num_commande = ${order_id}`;
29                xhr.send(JSON.stringify({sql: sql}));
30            }
31
32            xhr = makeRequest();
33
34            sql = `UPDATE commande SET etat_commande = 2, date_traitement = '${date}' WHERE num_commande = ${order_id}`;
35            xhr.send(JSON.stringify({sql: sql}));
36        }
37    }
38    window.location.reload();
39 }
```

Ce script PHP est utilisé par la fonction JavaScript pour gérer les stocks et traiter les commandes.

```
1 <?php
2 $servername = "localhost";
3 $username = "root";
4 $password = "";
5 $dbname = "site_e-commerce";
6
7 $sql = json_decode(file_get_contents("php://input"))->sql;
8
9 $link = mysqli_connect($servername, $username, $password, $dbname);
10 $result = mysqli_query($link, $sql);
11
12 mysqli_close($link);
13 ?>
```

## K – Conclusion

L'interface utilisateur (IHM) permet au préparateur de commande de peser et d'imprimer facilement un code-barres pour chaque produit commandé. Le site web, quant à lui, permet au préparateur de voir les produits commandés, de traiter ces commandes et de gérer les stocks des produits.

### III - Étudiant 3: GROSHEY Thibaut

#### A - Module de Facturation

**Facture**

|                       |   |   |
|-----------------------|---|---|
| De                    | A |   |
| MarketMate            |   | Nom du client <input type="text" value="thibaut"/>              |
| MarketMate@email.com  |   | client@email.com <input type="text" value="thibaut@gmail.com"/> |
| 3 rue Albert Samain   |   | rue <input type="text" value="8 rue des pommier"/>              |
| telephone: 0125262898 |   | telephone: <input type="text" value="0614256897"/>              |

|                     |   |                        |
|---------------------|---|------------------------|
| numéro de facture : | <input type="text" value="11044"/>      | numero facture et date |
| Date :              | <input type="text" value="2023-03-27"/> |                        |

| nom article  | cout unitaire | quantite | montant |
|--------------|---------------|----------|---------|
| 1 Oignons    | 2 €/kg        | 1500 g   | 3 €     |
| 2 Tomates    | 3 €/kg        | 3000 g   | 9 €     |
| 3 Concombres | 2 €/kg        | 2000 g   | 4 €     |

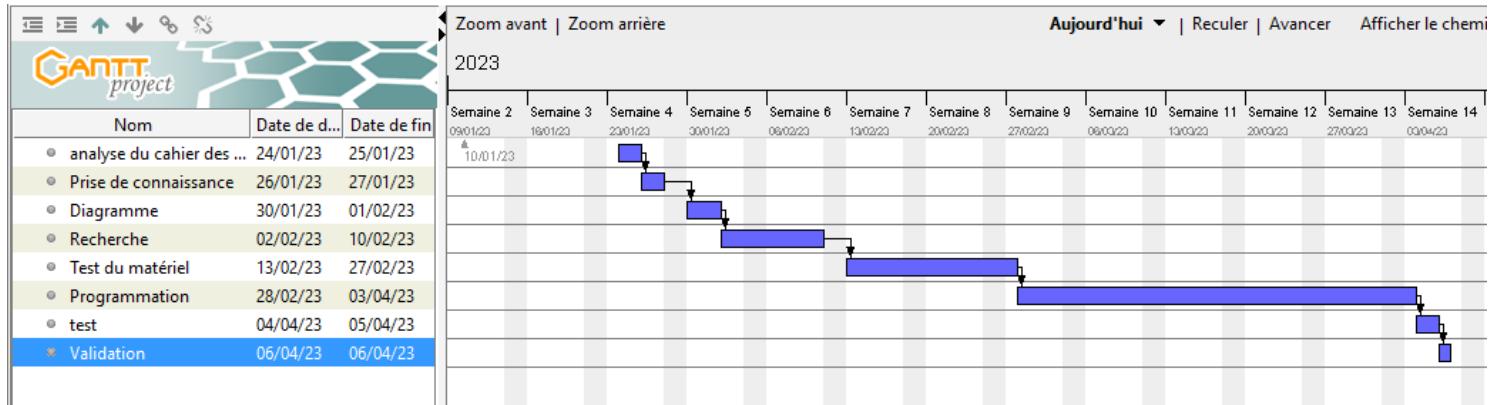
<  >

|              |                      |                        |
|--------------|----------------------|------------------------|
| code barre : | <input type="text"/> | affichage des produits |
|--------------|----------------------|------------------------|

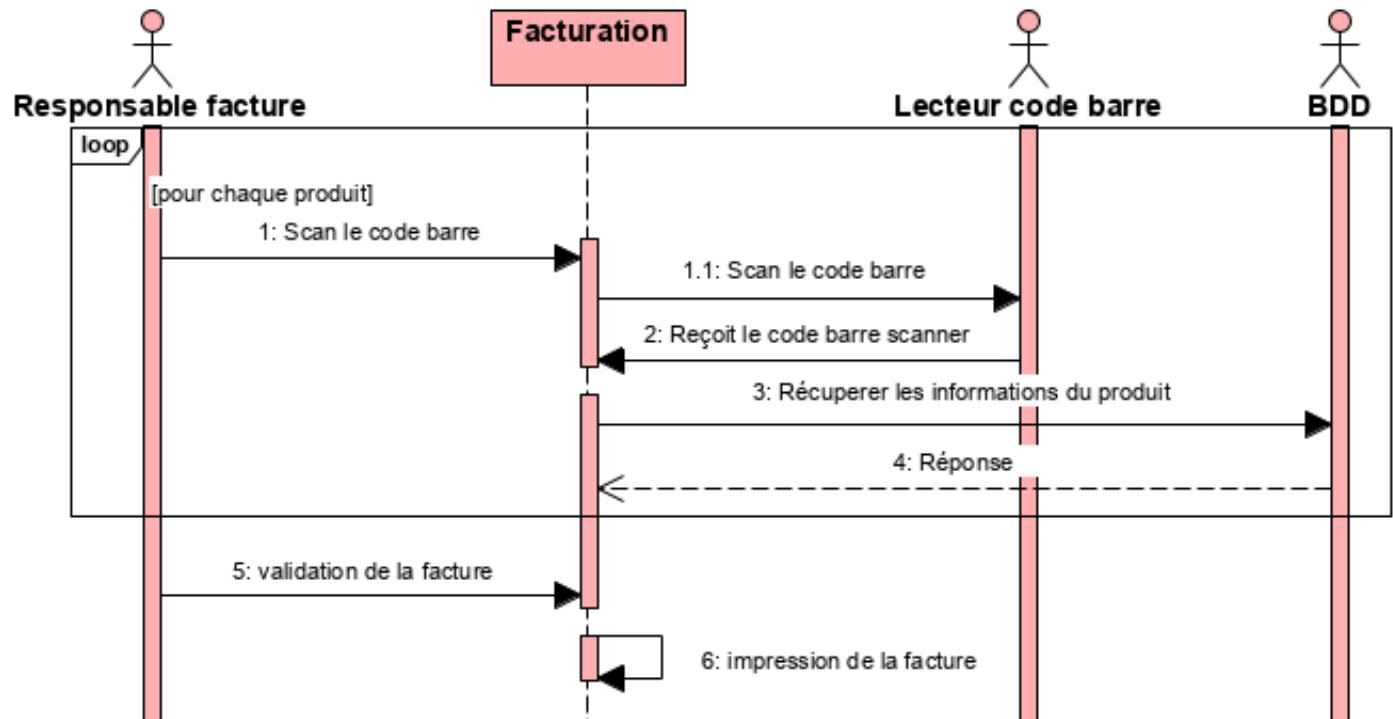
  

|                 |                                   |              |
|-----------------|-----------------------------------|--------------|
| Montant total : | <input type="text" value="16 €"/> | Save facture |
|-----------------|-----------------------------------|--------------|

## B - Planification :



## C - Diagramme séquence :

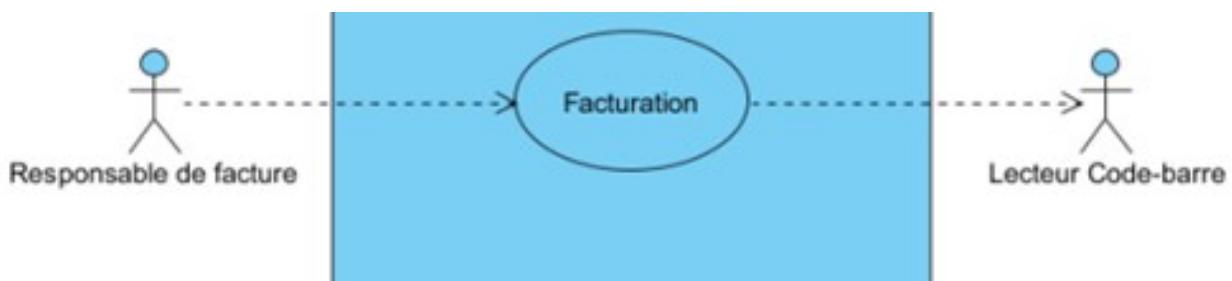


Voici le diagramme de classe de la partie facturation on y retrouve, le responsable facture qui pour chaque produit va scanner le code barre sur le module de facturation à l'aide du lecteur de code barre, celui-ci va donc envoyer le code barre sur le module. Ce module va donc récupérer les informations du produit sur la base de données et celle-ci va donc répondre à cette demande. Pour finir le responsable facture va donc valider la facture si la facture est validée celle-ci est imprimée.

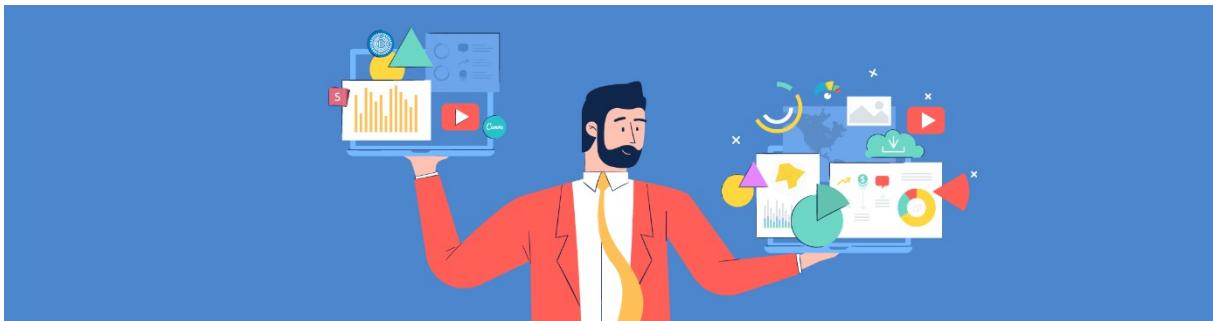
## D - Diagramme déploiement :



## E - Diagramme cas d'utilisation :



## F - Présentation de ce Module



Ce Module de facturation nous sera utile pour le projet e-commerce car il permet justement la création et l'impression de la facture pour le client.

Il est notamment en relation avec le module de création des codes barre, qui va être utile pour la création de la facture à l'aide des codes-barres générés, et aussi grâce à la base de données créées par le module de création du site web, de plus il est relié au module de paiement.

## G - Présentation du matériel et du logiciel utilisé

Pour ce module j'ai donc utilisé un lecteur code barre scan plus 1500 de la marque Datalogic :



Le logiciel utilisé pour la création de ce module est QT Creator :



Qt Creator est un environnement de développement intégré multiplate-forme, faisant partie du Framework Qt. Il est donc orienté pour la programmation en C/C++.

## H - Fonctionnement du lecteur code barre avec l'infrarouge



Le lecteur de code-barre infrarouge fonctionne en utilisant des diodes électroluminescentes (DEL), qui émettent des faisceaux de lumières infrarouge sur le code-barre. Le code-barre réfléchit, ensuite cette lumière infrarouge et les capteurs du lecteur détectent les modulations de la lumière pour déchiffrer le code-barre.

Le processus commence lorsque le lecteur de code-barre est activé et que l'utilisateur le pointe vers le code-barre. Le lecteur utilise alors ses DEL pour émettre un faisceau de lumière infrarouge sur le code-barre. Le code-barre réfléchit la lumière infrarouge, qui est détectée par les capteurs du lecteur.

Les capteurs convertissent ensuite les modulations de la lumière réfléchie en signaux électriques, qui sont envoyés à l'unité centrale de traitement du lecteur. Cette unité centrale de traitement utilise ensuite des algorithmes, pour interpréter les signaux et déchiffrer le code-barre. Le code-barre peut alors être affiché à l'écran ou stocké dans la mémoire du lecteur.

Dans le cadre de notre projet c'est grâce à l'IHM que l'on déchiffre ce code barre.

## I - Liste des différents codes barres et explication du code barre EAN13

Il existe plusieurs types de codes-barres, chacun ayant une structure et une utilisation spécifiques. Voici une liste des principaux types de codes-barres :

Code-barre EAN : Ce type de code-barre est utilisé en Europe et dans de nombreux autres pays pour la vente au détail. Il est généralement représenté sous forme de code-barre EAN-13 et est constitué de 13 chiffres.

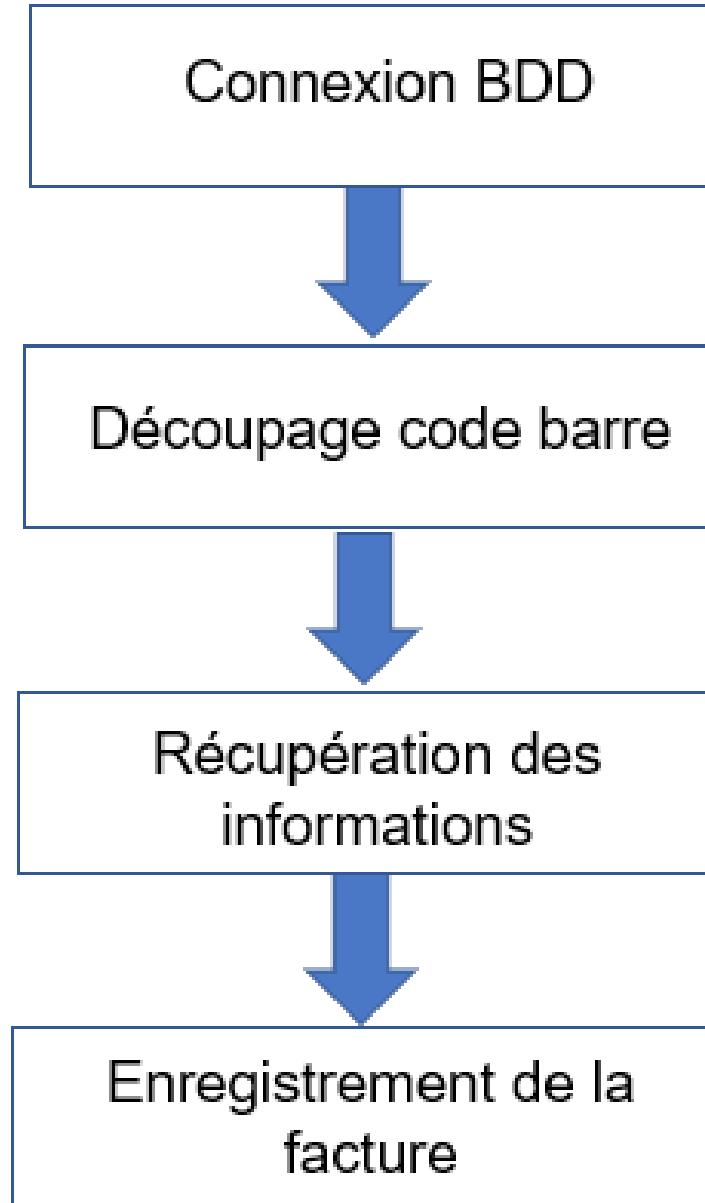
Code-barre Code 39 : Ce type de code-barre est largement utilisé dans l'industrie, pour l'identification des produits et des emplacements. Il peut contenir des lettres, des chiffres et certains symboles.

Code-barre Code 128 : Ce type de code-barre est également utilisé dans l'industrie, pour l'identification des produits et des emplacements. Il peut contenir un grand nombre de caractères, y compris des lettres, des chiffres et des symboles.

Maintenant, pour expliquer le code-barre EAN-13 qui est utilisé dans notre projet, c'est un type de code-barre utilisé pour la vente au détail en Europe. Il est constitué de 13 chiffres et chaque chiffre représente une information spécifique sur le produit. Les deux premiers chiffres représentent le code du pays où le produit a été fabriqué, les cinq chiffres suivants représentent le code de l'entreprise qui a fabriquée le produit, et les cinq derniers chiffres représentent le code du produit spécifique.

Le code-barre EAN-13 est généralement représenté sous forme de lignes noires et blanches d'épaisseurs différentes, qui représentent les chiffres du code. Le lecteur de code-barre utilise un laser pour détecter les changements de réflectivité des lignes noires et blanches du code-barre, et il les traduit ensuite en chiffres qui peuvent être interprétés par un ordinateur.

J - Organigramme des étapes du programme



## K - Connexion à la Base de données

```
db = QSqlDatabase::addDatabase("QODBC");
// set the database connection parameters
db.setHostName("localhost");
db.setDatabaseName("site_e-commerce");
db.setUserName("root");
db.setPassword("");
// db.setConnectOptions("MySQL ODBC 8.0 unicode Driver");

// open the database connection
if (!db.open()) qDebug() << "Failed to connect to database:" << db.lastError().text();
else qDebug() << "Connexion réussite";
```

Le code fourni permet de se connecter à une base de données MySQL en utilisant la bibliothèque Qt. Il est écrit en C++ et comporte plusieurs lignes de code qui configurent les paramètres de connexion de la base de données.

La première ligne de code crée un objet QSqlDatabase appelé "db" et configure le pilote de la base de données à utiliser. Dans notre cas, le pilote est "QODBC", ce qui indique que la base de données utilisée est MySQL.

Les lignes suivantes définissent les paramètres de connexion tels que le nom d'hôte, le nom de la base de données, le nom d'utilisateur et le mot de passe.

La dernière partie du code teste la connexion à la base de données en utilisant la méthode "open()" de l'objet QSqlDatabase. Si la connexion réussie, un message de réussite est affiché dans la console, sinon un message d'erreur est affiché avec une description de l'erreur.

Le code permet donc la connexion à la base de données site\_e-commerce qui possède les tables suivantes :

## L - Découpage du code barre

```
void MainWindow::on_pushButton_produit_clicked()
{
    QString codebarre = ui->textEdit_CodeBarre->toPlainText();
    QStringList codebarreList;
    int startIndex = 0;
    int chunkSize = 6;

    while (startIndex < codebarre.length()) {
        QString chunk = codebarre.mid(startIndex, chunkSize);
        codebarreList.append(chunk);
        startIndex += chunkSize;
    }
}
```

Le code ci-dessus permet de diviser une chaîne de caractères qui représente un code-barre en segments de taille fixe et de stocker chaque segment dans une liste de chaînes de caractères. Cette méthode est utilisée pour faciliter le traitement et la manipulation du code-barre.

La première ligne récupère le contenu d'un widget QTextEdit nommé "textEdit\_Code-Barre" et le stocke dans une variable QString nommée "codebarre".

Ensuite, une liste vide de chaînes de caractères nommée "codebarreList" est créée. Cette liste sera utilisée pour stocker les segments du code-barre.

La taille de chaque segment est définie à 6 caractères, et une boucle while est initiée pour extraire chaque segment de la chaîne de caractères représentant le code-barre.

Dans la boucle, la méthode "mid" de la classe QString est utilisée pour extraire un segment de la chaîne de caractères en fonction de l'index de début et de la taille du segment. Le segment extrait est ensuite ajouté à la liste "codebarreList" à l'aide de la méthode "append".

Enfin, l'index de début est incrémenté de la taille du segment traité, ce qui permet à la boucle de passer au segment suivant.

```

codebarreList[0] = codebarreList[0].remove(QRegExp("^[0]*"));
codebarreList[1] = codebarreList[1].remove(QRegExp("^[0]*")); //sert à retirer les 0 devant un chiffre supérieur à 0
codebarreList[1] = codebarreList[1].remove(codebarreList[1].size() - 1, 1); // sert à enlever le dernier chiffre
ui->textEdit_CodeBarre->clear();
qDebug() << codebarreList[0] << endl;
qDebug() << codebarreList[1] << endl;

```

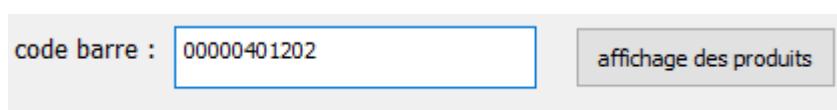
Le code ci-dessus manipule une liste de chaînes de caractères qui représentent des codes-barres.

Les premières lignes de code utilisent la méthode "remove()" pour supprimer les zéros en début de chaîne de la liste "codebarreList".

Les zéros en début de code-barre sont souvent ajoutés pour s'assurer que le code-barre a la bonne longueur, mais ils ne sont pas nécessaires pour identifier le produit. En les supprimant, le code permet d'obtenir la valeur réelle du code-barre qui est utilisé pour identifier le produit.

La troisième ligne de code utilise également la méthode "remove()", mais cette fois-ci, elle supprime le dernier caractère de la deuxième chaîne de caractères de la liste "codebarreList". Cette étape est nécessaire car le dernier caractère d'un code-barre représente le chiffre de contrôle et n'est pas utilisé pour identifier le produit. Le chiffre de contrôle est utilisé pour vérifier l'exactitude du code-barre lors de la numérisation et de la lecture du code, mais il ne fait pas partie de la réelle valeur du code-barre.

Le code barre scanné est donc affiché sur cette ligne de l'IHM :



Une fois le bouton sélectionné le code sera effectué pour découper ce code barre.

## M - Récupération des informations de la BDD

```
QSqlQuery requete;
if(requete.exec("SELECT * FROM produit WHERE id = " + codebarreList[0])) {
    qDebug() << "Ok - requete";

    // Boucle qui permet de parcourir les enregistrements renvoyés par la requête
    while(requete.next()) {
```

Le code exécute une requête SQL sur une base de données et récupère les informations du produit correspondant à un identifiant spécifié. Pour cela, il utilise une instance de la classe QSqlQuery qui permet d'exécuter des requêtes SQL sur la base de données.

La requête SQL est construite en utilisant la concaténation de la chaîne de caractères "SELECT \* FROM produit WHERE id = " avec l'identifiant spécifié dans la première chaîne de caractères de la liste "codebarreList". Cette requête SQL est ensuite exécutée à l'aide de la méthode "exec" de l'objet QSqlQuery.

Si l'exécution de la requête SQL est réussie, un message de débogage est affiché dans la console. L'instruction "if" permet de vérifier si la requête a été exécutée avec succès ou non.

De plus, ce code utilise la classe QSqlQuery pour exécuter une requête SQL sur une base de données et récupérer les informations d'un produit à partir d'un identifiant spécifié.

Les informations sont récupérées dans cette table produit :

|                          | id                        | nom_produit       | prix_unitaire | poids | stock | type             | url  |  |
|--------------------------|---------------------------|-------------------|---------------|-------|-------|------------------|--|--|
| <input type="checkbox"/> | Éditer  Copier  Supprimer | 1 Concombre       |               | 1     | 1     | 200 périssable   | https://static.wikia.nocookie.net/house-party/images/... |  |
| <input type="checkbox"/> | Éditer  Copier  Supprimer | 2 Tomate          |               | 1     | 1     | 200 périssable   | https://pngimg.com/d/tomato_PNG12511.png                 |  |
| <input type="checkbox"/> | Éditer  Copier  Supprimer | 3 Oignon          |               | 1     | 1     | 200 périssable   | https://pngimg.com/d/onion_PNG99190.png                  |  |
| <input type="checkbox"/> | Éditer  Copier  Supprimer | 4 Carotte         |               | 1     | 1     | 200 périssable   | https://www.transparentpng.com/thumb/carrot/AciY35...    |  |
| <input type="checkbox"/> | Éditer  Copier  Supprimer | 5 Pommes de terre |               | 1     | 1     | 200 périssable   | https://www.lespommesdeterre.com/wp-content/themes...    |  |
| <input type="checkbox"/> | Éditer  Copier  Supprimer | 6 Pomme           |               | 1     | 1     | 200 périssable   | https://freepngimg.com/save/4495-apple-png-image/5...    |  |
| <input type="checkbox"/> | Éditer  Copier  Supprimer | 7 kiwi            |               | 4     | 1     | 10000 périssable | https://static.vecteezy.com/system/resources/previ...    |  |

```
QString id = (requete.value("NOM_PRODUIT").toString());  
/ui->textEdit->append(id);  
ui->tableWidget_data->insertRow(0);  
ui->tableWidget_data->setItem(0, 0, new QTableWidgetItem(id));
```

Les lignes de code ci-dessus permettent d'ajouter une nouvelle ligne dans un tableau nommé "tableWidget\_data" et d'y insérer une valeur récupérée d'une requête SQL précédemment exécutée. Plus précisément, la première ligne de code insère une nouvelle ligne vide dans le tableau "tableWidget\_data" à l'index 0, tandis que la deuxième ligne de code créer un nouvel objet QTableWidgetItem avec la valeur récupérée de la colonne "NOM\_PRODUIT" de la requête SQL précédemment exécutée et l'ajoute à la première cellule de cette nouvelle ligne.

Ainsi, le code permet de récupérer une valeur de la colonne "NOM\_PRODUIT" et de l'ajouter dans une nouvelle ligne du tableau "tableWidget\_data" pour afficher cette information dans l'interface utilisateur de l'application.

Ce code est donc affiché comme ceci dans l'IHM :

|   | nom article |
|---|-------------|
| 1 | Concombre   |
| 2 | Oignon      |
| 3 | Pomme       |
| 4 | Carotte     |

```

int num = poids.toInt();
float num2 = num;
QString montant = QString::number(prix.toInt() * (num2 / 1000));
ui->tableWidget_data->setItem(0, 3, new QTableWidgetItem(montant + " €"));

```

Le code effectue des calculs pour déterminer le montant total d'un produit, en utilisant son poids et son prix. Tout d'abord, la chaîne de caractères "poids" est convertie en un entier à l'aide de la méthode "toInt" de la classe `QString`, qui stocke le résultat dans la variable "num". Ensuite, la valeur entière est stockée dans "num" et est convertie en un nombre à virgule et est stockée dans la variable "num2".

Ensuite, le prix du produit est également converti en un entier en utilisant la méthode "toInt", puis multiplié par la valeur de "num2" divisée par 1000 pour obtenir le montant total en euros. Le montant total est stocké dans la variable "montant".

Enfin, la valeur de "montant" est ajoutée à la troisième cellule de la première ligne de l'objet `QTableWidget` "tableWidget\_data", suivie du symbole "€" pour indiquer la devise.

Tout cela est donc affiché dans un tableau :

|   | nom article | cout unitaire | quantite | montant  |
|---|-------------|---------------|----------|----------|
| 1 | Concombre   | 1 €/kg        | 4152 g   | 4.152 €  |
| 2 | Oignon      | 1 €/kg        | 512 g    | 0.512 €  |
| 3 | Pomme       | 1 €/kg        | 512 g    | 0.512 €  |
| 4 | Carotte     | 1 €/kg        | 14052 g  | 14.052 € |
|   |             |               |          |          |

## N - Récupération des informations client

```
id = arg1;//recupere le texte
QString username = "";
QString email = "";

QSqlQuery requete3;
if(requete3.exec("SELECT * FROM commande WHERE num_commande = " + id)) {
    qDebug() << "Ok - requete";

    // Boucle qui permet de parcourir les enregistrements renvoyés par la requête
    while(requete3.next()) {
        id = (requete3.value("users_id")).toString();
    }
}

if(requete3.exec("SELECT * FROM client WHERE id = " + id))
{
    while(requete3.next())
    {
        username = requete3.value("username").toString();
        email = requete3.value("email").toString();
    }
}

ui->lineEdit_username->setText(username);
ui->lineEdit_email->setText(email);
```

Ce code récupère le numéro de commande dans une variable appelée "id", puis exécute une requête SQL pour récupérer le nom d'utilisateur et l'e-mail associés à ce numéro de commande à partir des tables "commande" et "client".

Voici la table client :

| <input type="checkbox"/> | Éditer | Copier | Supprimer | 1 | Githendra Perera | githendra.perera@gmail.com | qmskldjfjkmlsdjfmkjlsqdfjksdfjkqmqsdf | 621548798 2023-04-13 17:18:52 |
|--------------------------|--------|--------|-----------|---|------------------|----------------------------|---------------------------------------|-------------------------------|
| <input type="checkbox"/> | Éditer | Copier | Supprimer | 2 | Lucas Grolleau   | lucas.grolleau@gmail.com   | qsdkjdhqskldjhfhqhklsdfhjkqsdjhklf    | 521653221 2023-04-13 17:18:52 |
| <input type="checkbox"/> | Éditer | Copier | Supprimer | 3 | Thibaut Grosheny | thibaut.grosheny@gmail.com | sqdfjkhqsdflhjkqshdfhjkqlsfdkjhl      | 187549865 2023-04-13 17:18:52 |

Tout cocher   Avec la sélection : Éditer Copier Supprimer Exporter

Tout d'abord, une requête SQL est exécutée sur la table "commande" pour récupérer l'ID d'utilisateur associé au numéro de commande. Ensuite, une deuxième requête SQL est exécutée sur la table "client" pour récupérer les informations d'utilisateur correspondantes en utilisant l'ID d'utilisateur obtenu à partir de la première requête.

Finalement, les informations de nom d'utilisateur et d'e-mail sont affichées dans deux champs de texte appelés "lineEdit\_username" et "lineEdit\_email".

|                  |   |
|------------------|---|
| Nom du client    | <input type="text" value="Githendra Perera"/>           |
| client@email.com | <input type="text" value="githendra.perera@gmail.com"/> |

## O - Enregistrement de la facture

```
void MainWindow::on_pushButton_save_clicked()
{
    QString fileName = QFileDialog::getSaveFileName(this, "Save to PDF", "", "*.pdf");
    if (fileName.isEmpty()) return;
    if (!fileName.endsWith(".pdf")) fileName += ".pdf";

    QPrinter printer(QPrinter::PrinterResolution);
    printer.setOutputFormat(QPrinter::PdfFormat);
    printer.setOutputFileName(fileName);

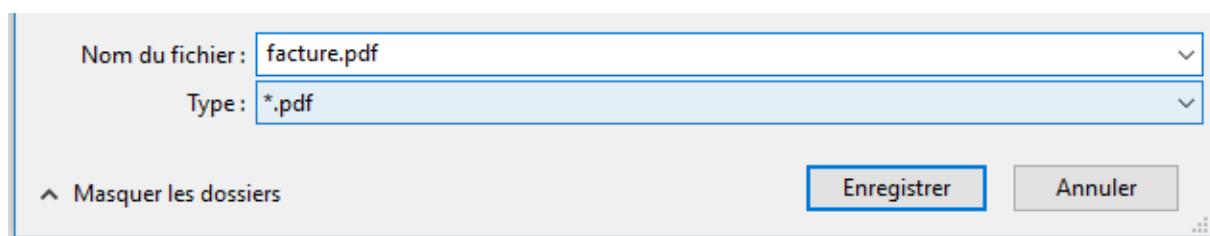
    QPainter painter;
    painter.begin(&printer);
    render(&painter);
    painter.end();
}
```

On utilise une boîte de dialogue pour permettre à l'utilisateur de sélectionner l'emplacement et le nom de fichier pour sauvegarder un fichier au format PDF.

La méthode "QFileDialog::getSaveFileName()" est appelée pour afficher la boîte de dialogue. Les paramètres de la méthode incluent :

- "this", qui est un pointeur vers la fenêtre parente de la boîte de dialogue
- "Save to PDF", qui est le titre de la boîte de dialogue
- une chaîne vide pour le répertoire initial
- "\*.pdf" pour le filtre de fichier.

Si l'utilisateur sélectionne un emplacement et un nom de fichier, le chemin complet du fichier est stocké dans la variable "fileName". Ensuite, une vérification est effectuée pour s'assurer que le nom de fichier se termine par l'extension ".pdf". Si ce n'est pas le cas, l'extension est ajoutée au nom de fichier.



Enfin, le nom de fichier est affiché à l'aide de la méthode "qDebug()" de la classe Qt.

```
"G:/build-lecteur_code_barre_v2-Desktop_Qt_5_9_6_MinGW_32bit-Debug/facture.pdf"
```

## P - Impression de la facture

```
void MainWindow::on_pushButton_imp_clicked()
{
    // Créer une instance de QPrinter
    QPrinter printer;

    // Ouvrir une boîte de dialogue pour choisir l'imprimante et les paramètres d'impression
    QPrintDialog printDialog(&printer, this);
    if (printDialog.exec() == QDialog::Accepted) {
        // Créer un QPainter pour dessiner sur la page imprimée
        QPainter painter(&printer);

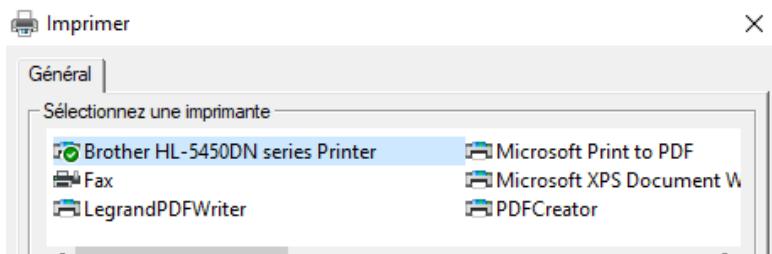
        // Dessiner l'interface sur la page
        ui->centralWidget->render(&painter);

        // Terminer l'impression
        painter.end();
    }
}
```

Le codes ci-dessus permet d'effectuer l'impression d'un fichier texte, tout d'abord, une instance de la classe QPrinter est créée, qui représente l'imprimante utilisée pour l'impression. Ensuite un objet QFile est créé pour le fichier à imprimer dont le nom de fichier est spécifié dans la variable fileName.

Si l'ouverture du fichier échoue alors un message d'erreur est affiché. Une boîte de dialogue est ensuite créée pour permettre à l'utilisateur de choisir l'imprimante et les paramètres d'impressions.

Si l'utilisateur clique sur le bouton d'impression, le code est exécuté et il affiche ceci :

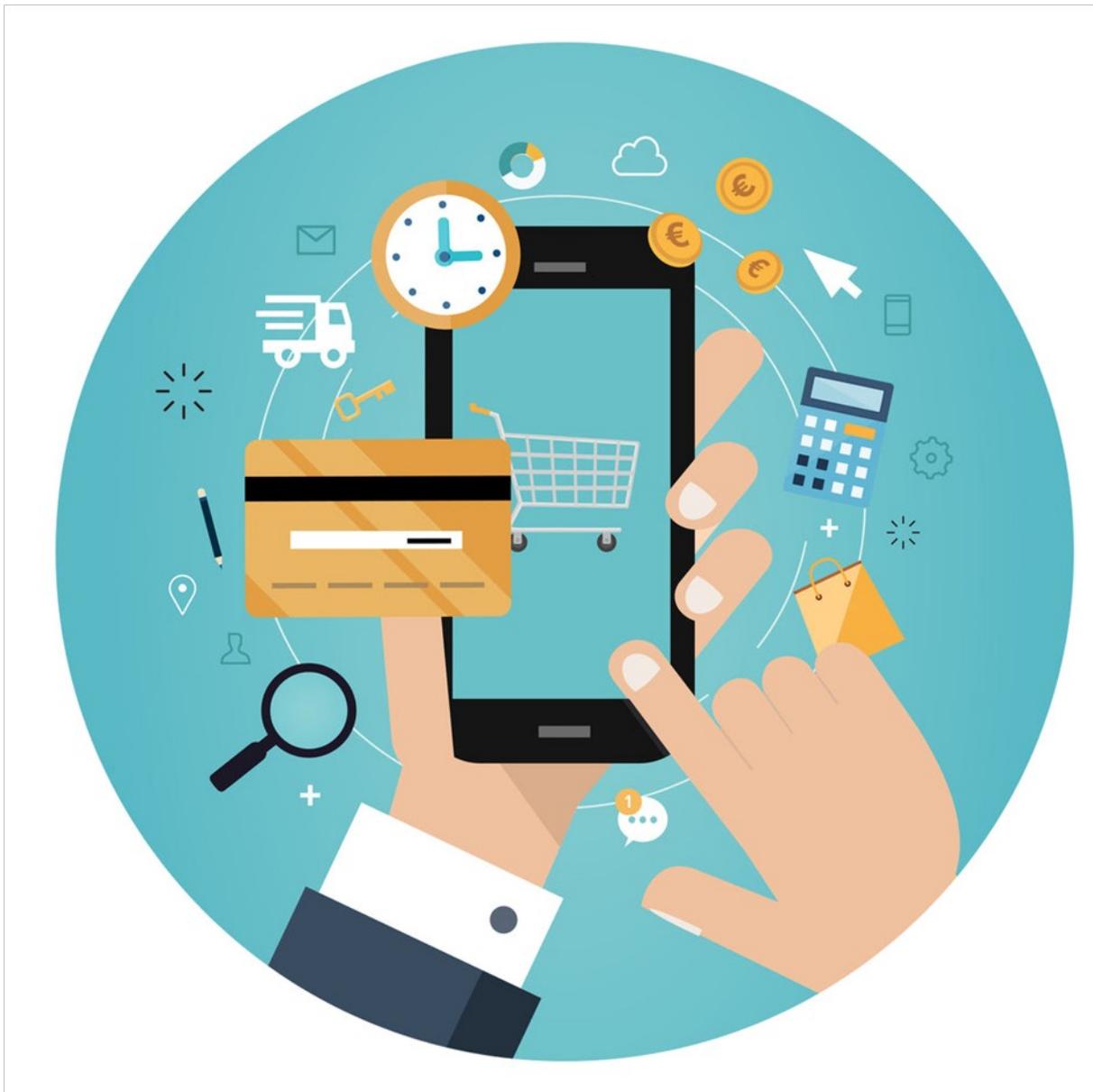


Un objet QPainter est créé pour dessiner sur la page imprimée, en passant l'objet QPrinter créé précédemment. Le contenu de l'interface utilisateur est ensuite dessiné sur la page à l'aide de la fonction « render() » de l'objet centralWidget, qui dessine l'ensemble du widget central sur le QPainter en utilisant les paramètres par défaut.

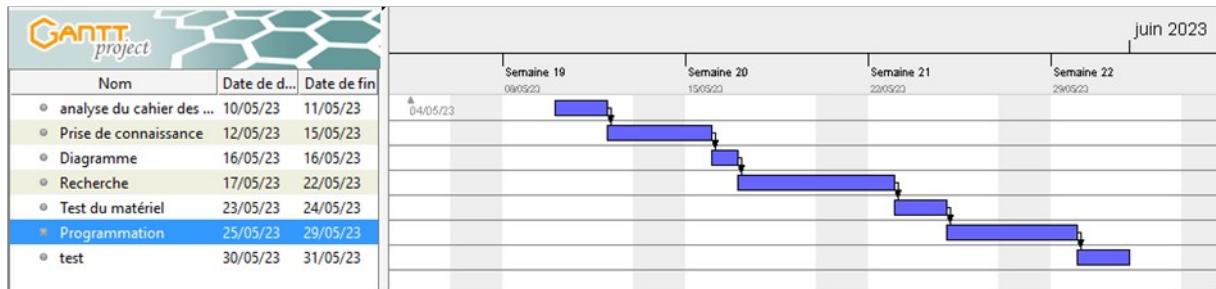
Enfin la fonction « end() » est appelée pour terminer l'impression. Si l'utilisateur annule l'impression alors la fonction se termine simplement, puis le fichier est fermé.

## IV- Étudiant 3: GROSHENY Thibaut Partie Bonus

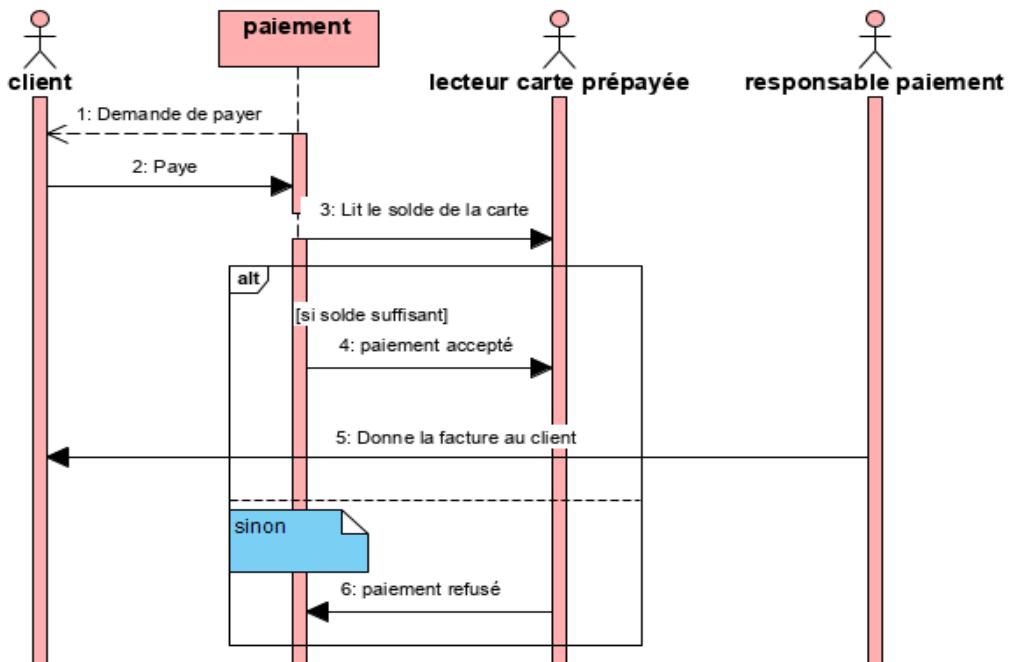
### A - Module Paiement



## B - Planification :

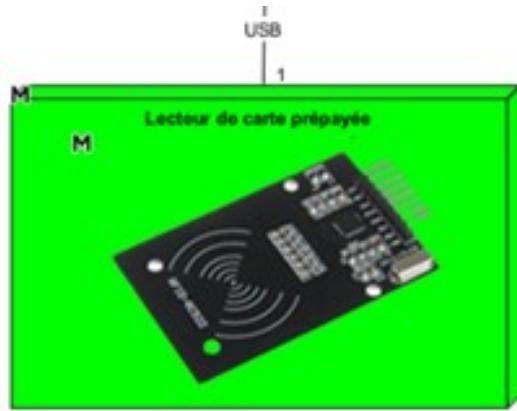


## C - Diagramme séquence :

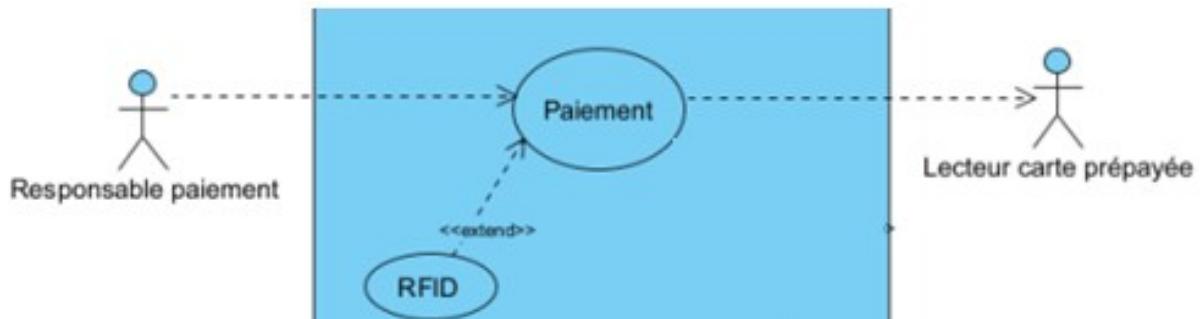


Voici le diagramme de classe on y retrouve, plusieurs acteurs tel que le client et le responsable paiement. Il y a aussi le lecteur de carte prépayer. Le programme de paiement demande au client de payer celui-ci va donc payer et par la suite il y a le lecteur qui lit le solde de la carte et si celui-ci est suffisant, le paiement est accepté et le responsable donne la facture au client sinon le paiement est refusé.

D - Diagramme de déploiement :



E - Diagramme cas d'utilisation :



## F - Présentation de ce Module



Ce Module de paiement nous sera utile pour le projet e-commerce car il permet justement le paiement du panier client

Il est notamment en relation avec le module de facturation qui va être utile pour connaître le prix que le client doit payer et pour ensuite lui remettre ces articles avec la facture imprimée.

Cette partie est un plus puisqu'un étudiant était absent durant tout le projet pour des raisons personnelles, j'ai donc tout de même rédigé le rapport et essayé de faire la partie qui lui était confiée pour le bon fonctionnement de notre projet.

## G - Présentation du matériel et du logiciel utilisé

Le logiciel utilisé pour la création de ce module est Arduino :



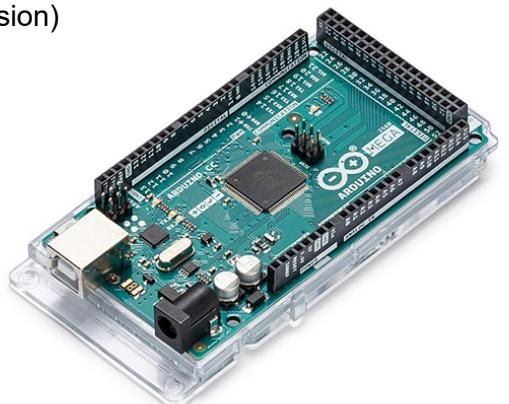
Arduino IDE est un logiciel open-source utilisé pour programmer les cartes électroniques Arduino. Il fournit un environnement convivial pour écrire, compiler et téléverser du code sur les cartes Arduino.

Le matériel utilisé pour ce module est :

Carte arduino mega :

La carte Arduino Mega utilise un microcontrôleur ATmega2560

- mémoire flash de 256 Ko
- 8 Ko de RAM
- 54 broches d'entrées/sorties numériques
- 15 utilisées comme sortie PWM (modulation de largeur d'impulsion)
- 16 entrées analogiques.



Lecteur MFRC522 :

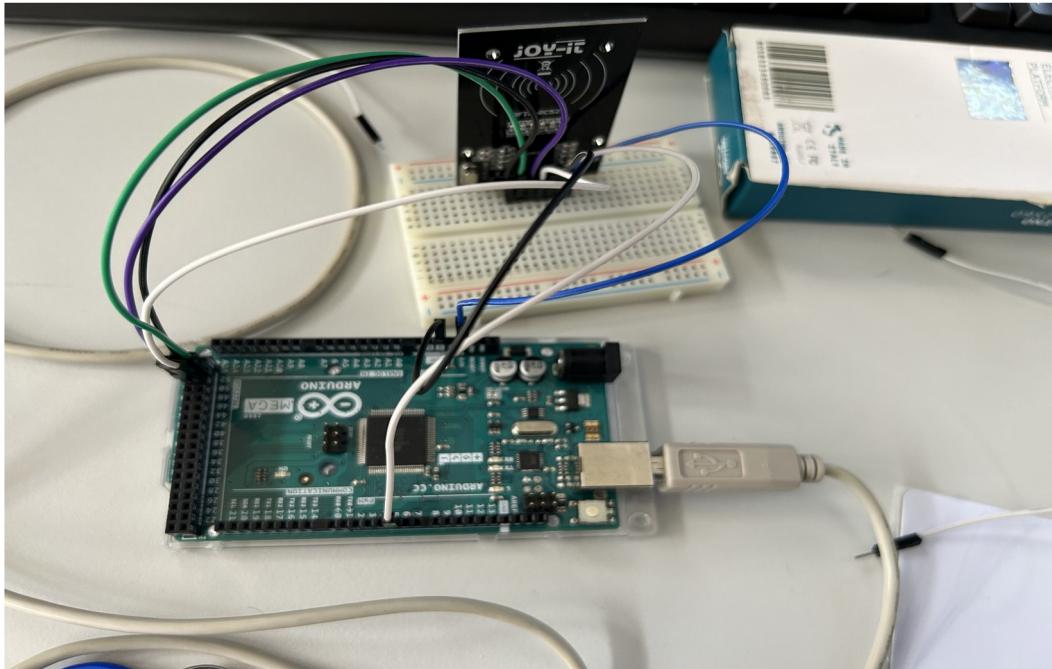


Le lecteur MFRC522 est un module RFID (Radio Frequency Identification) utilisé pour lire et écrire des tags RFID

Caractéristique du lecteur :

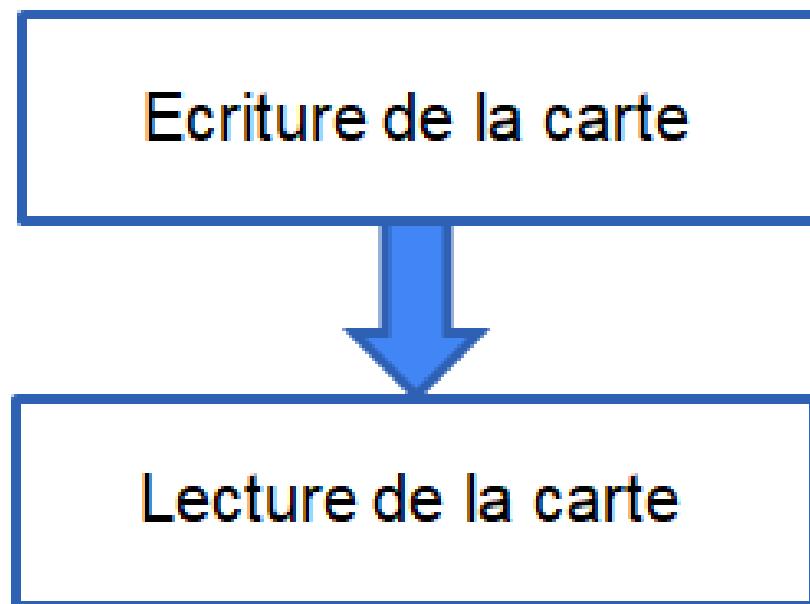
- Fréquence de 13,56 MHz
- Portée de lecture 3 à 5 centimètres

## H - Câblage du module



Voici le câblage du module de paiement donc on peut voir la carte Arduino et le lecteur MFRC522 qui est connecté ensemble grâce à la plaque labdec, ce schéma nous est utile pour l'écriture et la lecture de la carte ou du badge.

## I - Organigramme des étapes du programme



## J - Écriture sur la carte ou le badge

```
// Ask for personal data: montant
Serial.println(F("Entrez le montant de la carte, se terminant par #"));
len = Serial.readBytesUntil('#', (char*)buffer, 30); // read montant from serial
for (byte i = len; i < 30; i++) buffer[i] = ' '; // pad with spaces

block = 1;
status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, &key, &(mfrc522.uid));
if (status != MFRC522::STATUS_OK) {
    Serial.print(F("PCD_Authenticate() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
} else Serial.println(F("PCD_Authenticate() success: "));

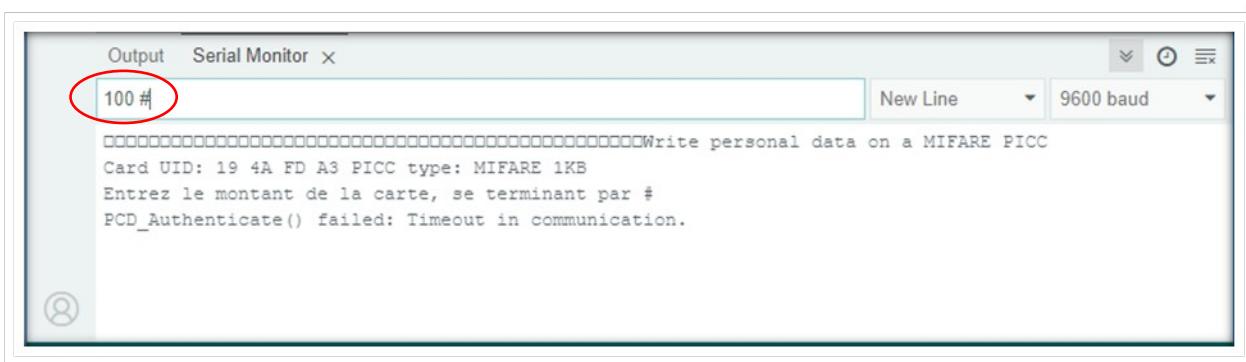
// Write block
status = mfrc522.MIFARE_Write(block, buffer, 16);
if (status != MFRC522::STATUS_OK) {
    Serial.print(F("MIFARE_Write() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
} else Serial.println(F("MIFARE_Write() success: "));
```

Le code ci-dessus effectue les opérations suivantes :

Cette ligne de code permet à l'utilisateur d'écrire le montant qui sera disponible sur la carte

```
Serial.println(F("Entrez le montant de la carte, se terminant par #"));
```

La syntaxe à respecter est donc la suivante :



Par la suite le code lit les données saisies par l'utilisateur jusqu'à ce que le caractère "#" soit rencontré et il les stocke dans le tableau buffer, grâce à cette ligne il stock la valeur :

```
len = Serial.readBytesUntil('#', (char*)buffer, 30);
```

La ligne suivante permet d'accéder à un block spécifique du lecteur

```
block = 1;  
status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, &key, &(mfrc522.uid));
```

S'il y a un échec dans cette opération, un message d'erreur est affiché, et le programme se termine. Sinon, un message de succès est affiché.

```
PCD_Authenticate() success:
```

Enfin cette ligne nous permet d'écrire les données stockées dans buffer dans le block authentifié du lecteur RFID :

```
status = mfrc522.MIFARE_Write(block, buffer, 16);
```

Si l'écriture échoue un message d'erreur est affiché et le programme se termine. Sinon un message de succès est affiché.

```
MIFARE_Write() success:
```

## K - Lecture de la carte ou du badge

```
Serial.print(F("Montant: "));  
  
byte buffer1[18];  
  
block = 4;  
len = 18;
```

Le code ci-dessus permet l'affichage de la chaîne de caractère "Montant : ", ensuite il y a la déclaration du tableau buffer1 et l'attribution de la valeur 4 à la variable block qui représente le block de mémoire à lire et l'attribution de la valeur 18 à la variable len qui représente la longueur des données à lire

```
status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, 4, &key, &(mfrc522.uid));  
if (status != MFRC522::STATUS_OK) {  
    Serial.print(F("Authentication failed: "));  
    Serial.println(mfrc522.GetStatusCodeName(status));  
    return;  
}  
  
status = mfrc522.MIFARE_Read(block, buffer1, &len);
```

Dans le code ci-dessus on retrouve une authentification avec la clé spécifiée (MFRC522::PICC\_CMD\_MF\_AUTH\_KEY\_A) pour accéder au block 4 du lecteur RFID si l'authentification échoue un message d'erreur est affiché et le programme se termine. La dernière ligne sert à lire les données du block 4 et il les stocks dans le tableau buffer1.

```
//PRINT montant  
for (uint8_t i = 0; i < 16; i++)  
{  
    if (buffer1[i] != 32)  
    {  
        Serial.write(buffer1[i]);  
    }  
}
```

Les lignes de code ci-dessus permettent l'affichage des données lues donc le montant en excluant les caractères d'espacements.

C'est donc pour cela que la ligne suivante est ajoutée au code pour justement ajouter des caractères d'espacements :

```
Serial.print(" ");
```

Une fois le code exécuté cela affiche le montant dans le moniteur série du logiciel Arduino

```
Card UID: 19 4A FD A3  
Card SAK: 08  
PICC type: MIFARE 1KB  
Montant: 00000000  
◆000 100  
**End Reading**
```

# Étudiant 1: ANNEXE

## index.php :

```
<?php
require '_header.php';

if (!isset($_SESSION['username'])) {
    // Rediriger vers la page de connexion
    header("Location: login.php");
    exit();
}

// Si le formulaire a été soumis, ajouter le produit au panier
if (isset($_POST['id']) && isset($_POST['quantite'])) {
    $id = $_POST['id'];
    $quantite = $_POST['quantite'];
    $produit = $DB->query('SELECT * FROM produit WHERE id = ?', array($id))[0];

    // Si le produit est périssable, convertir la quantité en kg
    if ($produit['type'] === "périssable") {
        $quantite = $quantite / 1000;
    }

    // Ajouter le produit au panier
    if (isset($_SESSION['panier'][$id])){
        $_SESSION['panier'][$id]['quantite'] += $quantite;
    } else {
        $_SESSION['panier'][$id] = array('quantite' => $quantite, 'produit' => $produit);
    }

    // Rediriger vers la page du panier
    header("Location: panier.php");
    exit();
}

// Récupérer les produits de la base de données
$produits = $DB->query('SELECT * FROM produit');
```

```

?>

<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>MarketMate</title>
    <link rel="stylesheet" type="text/css" href="home.css">
    <link rel="stylesheet"
        href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.2/css/all.min.css">
</head>
<body>
    <header class="header">
        <div class="header__logo">
            <a href="index.php"></a>
        </div>
        <h1 class="header__title">MarketMate</h1>
        <nav class="header__nav">
            <ul class="header__list">
                <li class="header__item"><a class="header__link"
href="index.php">Accueil</a></li>
                <li class="header__item"><a class="header__link"
href="#produits">Produits</a></li>
                <li class="header__item"><a class="header__link"
href="#promotions">Promotions</a></li>
                <li class="header__item"><a class="header__link" href="panier.php"><i class="fa
fa-shopping-cart"></i>Panier</a></li>
                    <?php if(isset($_SESSION['username'])): ?>
                    <li class="header__item"><a class="header__link" href="dashboard.php"><i
class="fa fa-user">&ampnbsp</i><?= ucfirst(htmlspecialchars($_SESSION['username'])) ?></a></li>
                <?php else: ?>
                    <li class="header__item"><a class="header__link" href="login.php"><i class="fa fa-
sign-in"></i> Se connecter</a></li>
                    <?php endif; ?>
            </ul>
        </nav>
    </header>

    <main class="main">

```

```

<?php foreach ($produits as $product): ?>
<div class="product">
    nom_produit; ?>">
    <h3 class="product__name"><?= $product->nom_produit; ?></h3>
    <p class="product__price"><?= number_format($product->prix_unitaire,2); ?>€<?= $product->unite; ?></p>
    <form method="POST" action="<?php if (isset($_SESSION['username'])) { echo 'add.php'; } else { echo 'login.php'; } ?>">
        <input type="hidden" name="id" value="<?= $product->id; ?>">
        <input type="number" name="quantite" min="1" max="100" value="1" onchange="updatePrice(this.value, <?= $product->prix_unitaire; ?>)">
        <p class="product__total-price"><?= number_format($product->prix_unitaire,2); ?>€</p>
        <?php
            if (isset($_SESSION['username'])) {
                // utilisateur connecté, ajouter le produit au panier
                echo '<button class="product__button" type="submit"><i class="fa fa-cart-plus"></i> Ajouter au panier</button>';
            } else {
                // utilisateur non connecté, rediriger vers la page de connexion
                echo '<button class="product__button" onclick="window.location.href=\\'login.php\\\'><i class="fa fa-cart-plus"></i> Connectez-vous pour ajouter au panier</button>';
            }
        ?>
    </form>
</div>
<?php endforeach; ?>
</main>
<script>
    function updatePrice(quantity, unitPrice) {
        const priceElement = event.target.nextElementSibling;
        const totalPrice = (quantity * unitPrice).toFixed(2);
        priceElement.innerHTML = ` ${totalPrice}€`;
    }
</script>

<footer class="footer">
    <p class="footer__copyright">© 2022 Produits frais</p>
</footer>

<script type="text/javascript"

```

```
src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js"></script>
<script type="text/javascript" src="app.js"></script>

</body>
</html>
```

### Functions.php :

```
<?php
require_once('config.php');

function register_user($email, $password) {
    global $db;

    // Vérifier si l'utilisateur existe déjà
    $query = "SELECT * FROM users WHERE email = '$email'";
    $result = mysqli_query($db, $query);
    if (mysqli_num_rows($result) > 0) {
        return 'Cet email est déjà utilisé';
    }

    // Hacher le mot de passe
    $hashed_password = password_hash($password, PASSWORD_DEFAULT);

    // Insérer l'utilisateur dans la base de données
    $query = "INSERT INTO users (email, password, created_at) VALUES ('$email',
'$hashed_password', NOW())";
    mysqli_query($db, $query);

    return 'Compte créé avec succès';
}

function login_user($email, $password) {
    global $db;

    // Récupérer l'utilisateur correspondant à l'email
    $query = "SELECT * FROM users WHERE email = '$email'";
    $result = mysqli_query($db, $query);
    if (mysqli_num_rows($result) == 0) {
        return 'Email ou mot de passe incorrect';
    }
```

```

$user = mysqli_fetch_assoc($result);

// Vérifier le mot de passe
if (!password_verify($password, $user['password'])) {
    return 'Email ou mot de passe incorrect';
}

// Démarrer la session et stocker l'ID de l'utilisateur
session_start();
$_SESSION['user_id'] = $user['id'];

return 'Connexion réussie';
}

function logout_user() {
    session_start();
    unset($_SESSION['user_id']);
    session_destroy();
    header('Location: index.php');
    exit();
}

function is_user_logged_in() {
    session_start();
    return isset($_SESSION['user_id']);
}

function get_logged_in_user() {
    global $db;

    session_start();

    if (!isset($_SESSION['user_id'])) {
        return false;
    }

    $user_id = $_SESSION['user_id'];
    $query = "SELECT * FROM users WHERE id = $user_id";
    $result = mysqli_query($db, $query);
    $user = mysqli_fetch_assoc($result);
}

```

```
    return $user;
}
?>
```

### Config.php :

```
<?php
$host = 'localhost';
$username = 'root';
$password = '';
$database = 'site_e-commerce';
$db = mysqli_connect($host, $username, $password, $database);
if (!$db) {
    die('Erreur de connexion: ' . mysqli_connect_error());
}

// Vérifier si l'utilisateur est connecté
session_start();
$userLoggedIn = false;
if(isset($_SESSION["loggedin"]) && $_SESSION["loggedin"] === true){
    $userLoggedIn = $_SESSION["id"];
}
?>
```

### Register.php :

```
<?php
require_once('includes/config.php');

if($_SERVER['REQUEST_METHOD'] == 'POST') {
    $username = trim($_POST['username']);
    $email = trim($_POST['email']);
    $password = trim($_POST['password']);
    $confirm_password = trim($_POST['confirm_password']);

    $errors = [];

    if(empty($username)) {
        $errors[] = 'Le champ "Nom d\'utilisateur" est obligatoire';
```

```

}

if(empty($email)) {
    $errors[] = 'Le champ "Email" est obligatoire';
} else {
    if(!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        $errors[] = 'Le format de l\'email n\'est pas valide';
    }
}

if(empty($password)) {
    $errors[] = 'Le champ "Mot de passe" est obligatoire';
} else {
    if(strlen($password) < 6) {
        $errors[] = 'Le mot de passe doit contenir au moins 6 caractères';
    }
    if($password !== $confirm_password) {
        $errors[] = 'Les mots de passe ne correspondent pas';
    }
}

if(empty($errors)) {
    $hashed_password = password_hash($password, PASSWORD_DEFAULT); // Hashage du mot de passe
    date_default_timezone_set('UTC');
    $date_creation = date('Y-m-d H:i:s');

    $stmt = $db->prepare("INSERT INTO client (username, email, password, date_creation)
VALUES (?, ?, ?, ?)");
    if (!$stmt) {
        die('Erreur de préparation de la requête : ' . $db->error);
    }
    $stmt->bind_param("ssss", $username, $email, $hashed_password, $date_creation);
    if (!$stmt->execute()) {
        die('Erreur d\'exécution de la requête : ' . $stmt->error);
    }
    // Récupérer l'id du client inséré
    $client_id = $stmt->insert_id;

    // Créer une nouvelle commande
    $date_commande = date('Y-m-d H:i:s');
}

```

```

$montant_commande = 0;
$etat_commande = 0;

$stmt = $db->prepare("INSERT INTO commande (date_commande,
montant_commande, etat_commande, client_id) VALUES (?, ?, ?, ?)");
if (!$stmt) {
    die('Erreur de préparation de la requête : ' . $db->error);
}
$stmt->bind_param("sdii", $date_commande, $montant_commande, $etat_commande,
$client_id);
if (!$stmt->execute()) {
    die('Erreur d\'exécution de la requête : ' . $stmt->error);
}

header('Location: login.php');
exit;
}
?>
```

```

<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Créer un compte</title>
        <link rel="stylesheet" href="register.css">
    </head>
    <body>
        <header class="header">
            <div class="header__logo">
                <a href="index.php"></a>
            </div>
            <h1 class="header__title">MarketMate</h1>
            <nav class="header__nav">
                <ul class="header__list">
                    <li class="header__item"><a class="header__link"
href="index.php">Accueil</a></li>
                    <li class="header__item"><a class="header__link"
href="#produits">Produits</a></li>
```

```

<li class="header__item"><a class="header__link"
href="#promotions">Promotions</a></li>
<li class="header__item"><a class="header__link" href="#panier">Panier</a></li>
<?php if(isset($_SESSION['username'])): ?>
<li class="header__item"><a class="header__link" href="dashboard.php"><i
class="fa fa-user"></i> Mon compte</a></li>
<li class="header__item"><a class="header__link" href="logout.php"><i class="fa
fa-sign-out"></i> Se déconnecter</a></li>
<?php else: ?>
<li class="header__item"><a class="header__link" href="login.php"><i class="fa fa-
sign-in"></i> Se connecter</a></li>
<?php endif; ?>
</ul>
</nav>

</header>
<div class="container">
<h1>Créer un compte</h1>

<?php if(!empty($errors)): ?>
<div class="errors">
<?php foreach($errors as $error): ?>
<p><?= $error ?></p>
<?php endforeach; ?>
</div>
<?php endif; ?>

<form method="post" action="">
<div class="form-group">
<label for="username">Nom d'utilisateur</label>
<input type="text" id="username" name="username" value="<?= $username ?? " ?
>" required>
</div>

<div class="form-group">
<label for="email">Email</label>
<input type="email" id="email" name="email" value="<?= $email ?? " ?>" required>
</div>

<div class="form-group">
<label for="password">Mot de passe</label>
<input type="password" id="password" name="password" required>

```

```

</div>

<div class="form-group">
    <label for="confirm_password">Confirmer le mot de passe</label>
    <input type="password" id="confirm_password" name="confirm_password"
required>
</div>

<button type="submit">Créer le compte</button>
</form>

<p>Déjà un compte? <a href="login.php">Se connecter</a></p>
</div>
</body>
</html>

```

### Login.php :

```

<?php
// Inclure le fichier de configuration et de connexion à la base de données
require_once('includes/config.php');

// Initialiser les variables pour stocker les informations d'identification de l'utilisateur
$username = "";
$password = "";
$username_err = "";
$password_err = "";

// Traitement du formulaire de connexion lors de la soumission
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Valider le nom d'utilisateur
    if (!isset($_POST["username"]) || strlen(trim($_POST["username"])) == 0) {
        $username_err = "Veuillez entrer votre nom d'utilisateur.";
    } else {
        $username = trim($_POST["username"]);
    }

    // Valider le mot de passe
    if (!isset($_POST["password"]) || strlen(trim($_POST["password"])) == 0) {
        $password_err = "Veuillez entrer votre mot de passe.";
    } else {

```

```

$password = trim($_POST["password"]);
}

// Vérifier les erreurs de saisie avant de continuer
if (empty($username_err) && empty($password_err)) {
    // Préparer une requête SELECT pour récupérer l'utilisateur correspondant aux
    informations d'identification fournies
    $sql = "SELECT id, username, password FROM client WHERE username = ?";

    if ($stmt = $db->prepare($sql)) {
        // Définir les paramètres et lier les variables à la requête préparée en tant que
        paramètres
        $param_username = $username;
        $stmt->bind_param("s", $param_username);

        // Exécuter la requête préparée
        if ($stmt->execute()) {
            // Stocker le résultat
            $stmt->store_result();

            // Vérifier si le nom d'utilisateur existe, si oui alors vérifier le mot de passe
            if ($stmt->num_rows == 1) {
                // Lier les résultats de la requête à des variables
                $stmt->bind_result($id, $username, $password_db);
                if ($stmt->fetch()) {
                    // Vérifier si le mot de passe saisi correspond au mot de passe dans la
                    base de données
                    if (password_verify($password, $password_db)) {
                        // Le mot de passe est correct, alors commencer une session
                        session_start();

                        // Stocker les données de l'utilisateur dans des variables de session
                        $_SESSION["loggedin"] = true;
                        $_SESSION["id"] = $id;
                        $_SESSION["username"] = $username;

                        // Rediriger l'utilisateur vers la page de tableau de bord
                        echo "<script>window.location.replace('index.php');</script>";
                    } else {
                        // Afficher un message d'erreur si le mot de passe est incorrect
                        $password_err = "Le mot de passe que vous avez entré n'est pas

```

```

valide.";

        }

    }

} else {
    // Afficher un message d'erreur si le nom d'utilisateur n

    // Afficher un message d'erreur si le nom d'utilisateur n'existe pas
    $username_err = "Aucun compte trouvé avec ce nom d'utilisateur.";

}

} else {
    echo "Oops! Quelque chose s'est mal passé. Veuillez réessayer plus tard.";
}

// Fermer la déclaration préparée
$stmt->close();
}

}

// Fermer la connexion à la base de données
$db->close();
}

?>

```

```

<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Connexion</title>
        <link rel="stylesheet" href="log.css">
        <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.2/css/all.min.css">
    <script>
        window.addEventListener("load", function() {
            document.querySelector(".container").classList.add("loaded");
       });
        const togglePassword = document.querySelector('.toggle-password');
        const passwordInput = document.querySelector('input[name="password"]');
        togglePassword.addEventListener('click', function () {
            const type = passwordInput.getAttribute('type') === 'password' ? 'text' :
            'password';

```

```

        passwordInput.setAttribute('type', type);
        this.querySelector('i').classList.toggle('fa-eye');
        this.querySelector('i').classList.toggle('fa-eye-slash');
    });
</script>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<script>
$(document).ready(function(){
    $('.toggle-password').click(function(){
        $(this).toggleClass('fa-eye fa-eye-slash');
        var input = $(this).parent().find('input');
        if(input.attr('type') == 'password') {
            input.attr('type', 'text');
        } else {
            input.attr('type', 'password');
        }
    });
});
</script>

</head>
<body>
    <header class="header">
        <div class="header__logo">
            <a href="index.php"></a>
        </div>
        <h1 class="header__title">MarketMate</h1>
        <nav class="header__nav">
            <ul class="header__list">
                <li class="header__item"><a class="header__link" href="index.php">Accueil</a></li>
                <li class="header__item"><a class="header__link" href="#produits">Produits</a></li>
                <li class="header__item"><a class="header__link" href="#promotions">Promotions</a></li>
                <li class="header__item"><a class="header__link" href="#panier">Panier</a></li>
                <?php if(isset($_SESSION['username'])): ?>
                <li class="header__item"><a class="header__link" href="dashboard.php"><i class="fa fa-user"></i> Mon compte</a></li>
            </ul>
        </nav>
    </header>
    <main>
        <div>
            <h2>Dashboard</h2>
            <p>Bienvenue sur votre tableau de bord MarketMate. Vous pouvez gérer vos produits, accéder à vos statistiques et paramètres de vente. N'oubliez pas de sauvegarder régulièrement vos modifications.</p>
            <ul>
                <li><a href="#">Produits </a></li>
                <li><a href="#">Statistiques </a></li>
                <li><a href="#">Paramètres </a></li>
            </ul>
        </div>
    </main>
</body>

```

```

<?php else: ?>
<li class="header__item"><a class="header__link" href="login.php"><i class="fa fa-sign-in"></i> Se connecter</a></li>
<?php endif; ?>
</ul>
</nav>

</header>
<div class="container">
<h1>Connexion</h1>
<p>Veuillez remplir ce formulaire pour vous connecter.</p>
<form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="post">
<div class="form-group <?php echo (!empty($username_err)) ? 'has-error' : '' ?>">
<label>Nom d'utilisateur ou adresse e-mail</label>
<input type="text" name="username" class="form-control" value="<?php echo htmlspecialchars($username); ?>">
<span class="help-block"><?php echo $username_err; ?></span>
</div>
<div class="form-group <?php echo (!empty($password_err)) ? 'has-error' : '' ?>">
<label>Mot de passe</label>
<div class="password-wrapper">
<input type="password" name="password" class="form-control">
<span class="toggle-password"><i class="fa fa-eye"></i></span>
</div>
<span class="help-block"><?php echo $password_err; ?></span>
</div>
<div class="form-group">
<input type="submit" class="btn btn-primary" value="Se connecter">
</div>
<div class="form-group">
<a href="forgot_password.php">Mot de passe oublié?</a>
</div>
<p>Vous n'avez pas de compte? <a href="register.php">S'inscrire maintenant</a>.</p>
</form>
</div>
</body>
</html>

```

## Dashboard.php :

```

<?php
require_once('includes/config.php');

// Vérifier si l'utilisateur est connecté, sinon rediriger vers la page de connexion
if(!$userLoggedIn) {
    header('Location: login.php');
    exit;
}

// Récupérer les informations de l'utilisateur connecté depuis la base de données
$stmt = $db->prepare("SELECT username, email FROM client WHERE id = ?");
$stmt->bind_param("i", $userLoggedIn);
$stmt->execute();
$result = $stmt->get_result();
if($result->num_rows === 0) exit('Aucun utilisateur trouvé');
$user = $result->fetch_assoc();

// Récupérer les commandes passées par l'utilisateur
$stmt = $db->prepare("SELECT * FROM commande WHERE client_id = ?");
$stmt->bind_param("i", $userLoggedIn);
$stmt->execute();
$result = $stmt->get_result();

?>
<!DOCTYPE html>
<html lang="fr">
    <head>
        <meta charset="UTF-8">
        <title>Panier</title>
        <link rel="stylesheet"
        href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.2/css/all.min.css">
        <link rel="stylesheet" type="text/css" href="dashboard.css">
        <meta name="viewport" content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    </head>
    <body>
        <header class="header">
            <div class="header__logo">
                <a href="index.php">
                    
    </a>
</div>
<h1 class="header__title">MarketMate</h1>
<nav class="header__nav">
    <ul class="header__list">
        <li class="header__item"><a class="header__link"
href="index.php">Accueil</a></li>
        <li class="header__item"><a class="header__link"
href="#produits">Produits</a></li>
        <li class="header__item"><a class="header__link"
href="#promotions">Promotions</a></li>
        <li class="header__item"><a class="header__link" href="panier.php"><i class="fa
fa-shopping-cart"></i> Panier</a></li>
        <?php if(isset($_SESSION['username'])): ?>
        <li class="header__item">
            <a class="header__link" href="dashboard.php">
                <i class="fa fa-user">&ampnbsp</i>
                <a href="dashboard.php" class="header__link"><?=
ucfirst(htmlspecialchars($_SESSION['username'])) ?></a>
            </a>
        </li>
        <?php else: ?>
        <li class="header__item">
            <a class="header__link" href="login.php">
                <i class="fa fa-sign-in"></i> Se connecter
            </a>
        </li>
        <?php endif; ?>
    </ul>
</nav>
</header>
<h1 class="titre-profil">Bienvenue sur votre profil, <?=
htmlspecialchars($user['username']) ?></h1>
<p class="adresse-email">Votre adresse e-mail : <?= htmlspecialchars($user['email']) ?></
p>

<div class="commandes">
    <?php
    // Récupérer les commandes avec un état de commande égal à 1
    $stmt = $db->prepare("SELECT * FROM commande WHERE client_id = ? AND
etat_commande = 1");

```

```

$stmt->bind_param("i", $userLoggedIn);
$stmt->execute();
$result = $stmt->get_result();
while($row = $result->fetch_assoc()): ?>
<div class="commande">
    <h2 class="commande-titre">Commande du <?= $row['date_commande'] ?></h2>
    <p class="commande-montant">Montant : <?= $row['montant_commande'] ?> €</p>
<?php
// Récupérer les articles pour cette commande
$stmt = $db->prepare("SELECT * FROM article WHERE num_commande = ? AND
client_id = ?");
$stmt->bind_param("ii", $row['num_commande'], $userLoggedIn);
$stmt->execute();
$articleResult = $stmt->get_result();
?>

<ul class='articles'>
<?php while($articleRow = $articleResult->fetch_assoc()): ?>
<?php
// Récupérer les informations sur le produit associé à cet article
$stmt = $db->prepare("SELECT * FROM produit WHERE id = ?");
$stmt->bind_param("i", $articleRow['produit_id']);
$stmt->execute();
$produitResult = $stmt->get_result();
$produitRow = $produitResult->fetch_assoc();
?>
<li class='article'>
    ">
    <div>
        <p><?= $produitRow['nom_produit'] ?> (<?= $articleRow['quantite'] ?> <?=
$produitRow['unite'] ?>)</p>
        <p><?= $articleRow['prix_total'] ?> €</p>
    </div>
</li>

<?php endwhile; ?>
</ul>
</div>
<?php endwhile; ?>
</div>
<p class="deconnexion"><a href="logout.php">Se déconnecter</a></p>

```

```
</body>
</html>
```

### Panier.php :

```
<?php
$dsn = 'mysql:host=localhost;dbname=site_e-commerce';
$username = 'root';
$password = "";

require_once "includes/config.php";
require_once "_header.php";

try {
    $pdo = new PDO($dsn, $username, $password);
} catch (PDOException $e) {
    echo 'Connexion échouée : ' . $e->getMessage();
}

function getProduitById($produit_id) {
    global $pdo;
    $stmt = $pdo->prepare("SELECT * FROM produit WHERE id = :produit_id");
    $stmt->bindParam(":produit_id", $produit_id, PDO::PARAM_INT);
    $stmt->execute();
    return $stmt->fetch(PDO::FETCH_ASSOC);
}

// Vérifie si l'utilisateur est connecté, sinon redirige vers la page de connexion
if(!isset($_SESSION["loggedin"]) || $_SESSION["loggedin"] !== true){
    header("location: login.php");
    exit;
}

// Récupérer l'id de l'utilisateur connecté
$client_id = $_SESSION["id"];

// Récupérer les informations du panier
$sql = "SELECT c.num_commande, c.date_commande, p.id AS produit_id,
```

```

p.nom_produit, p.prix_unitaire, a.quantite, p.url, (p.prix_unitaire * a.quantite) as prix_total
    FROM commande c
    JOIN article a ON c.num_commande = a.num_commande AND c.client_id =
a.client_id
    JOIN produit p ON a.produit_id = p.id
    WHERE c.client_id = :client_id AND c.etat_commande = 0";

if($stmt = $pdo->prepare($sql)){
    $stmt->bindParam(":client_id", $client_id, PDO::PARAM_INT);

    if($stmt->execute()){
        $rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
    } else{
        echo "Oops! Something went wrong. Please try again later.";
    }
}

// Calculer le montant total de la commande
$total = 0;
foreach($rows as $row){
    $prix_total = $row["prix_total"];
    $total += $prix_total;
}

// Envoyer le montant total à la table commande
$sql = "UPDATE commande SET montant_commande = :total WHERE client_id
= :client_id AND etat_commande = 0";
if($stmt = $pdo->prepare($sql)){
    $stmt->bindParam(":total", $total, PDO::PARAM_INT);
    $stmt->bindParam(":client_id", $client_id, PDO::PARAM_INT);
    if(!$stmt->execute()){
        echo "Oops! Something went wrong. Please try again later.";
    }
}

?>

<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">

```

```

<title>Panier</title>
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.2/css/all.min.css">
    <link rel="stylesheet" type="text/css" href="panier.css">
        <meta name="viewport" content="width=device-width, user-scalable=no, initial-
scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
            <link rel="stylesheet" href=".main.css">
</head>
<body>
    <header class="header">
        <div class="header__logo">
            <a href="index.php"></a>
        </div>
        <h1 class="header__title">MarketMate</h1>
        <nav class="header__nav">
            <ul class="header__list">
                <li class="header__item"><a class="header__link"
href="index.php">Accueil</a></li>
                <li class="header__item"><a class="header__link"
href="#produits">Produits</a></li>
                <li class="header__item"><a class="header__link"
href="#promotions">Promotions</a></li>
                <li class="header__item"><a class="header__link" href="panier.php"><i class="fa
fa-shopping-cart"></i> Panier</a></li>
                <?php if(isset($_SESSION['username'])): ?>
                    <li class="header__item"><a class="header__link" href="dashboard.php"><i
class="fa fa-user">&nbsp;</i><a href="dashboard.php" class="header__link"><?=
ucfirst(htmlspecialchars($_SESSION['username'])) ?></a></li>
                <?php else: ?>
                    <li class="header__item"><a class="header__link" href="login.php"><i class="fa fa-
sign-in"></i> Se connecter</a></li>
                <?php endif; ?>
            </ul>
        </nav>
    </header>

    <div class="wrapper">
        <h2>Panier</h2>
        <table class="table table-bordered">
            <thead>
                <tr>
                    <th style="text-align: center;">Images</th>

```

```

<th>Nom du produit</th>
<th>Prix unitaire</th>
<th>Quantité</th>
<th>Total</th>
<th>Supprimer</th>
</tr>
</thead>
<tbody>
<?php
$total = 0;
foreach($rows as $row){
    $prix_total = $row["prix_total"];
    $total += $prix_total;
    $produit = getProduitById($row["produit_id"]);
?>
<tr>
<td style="text-align: center;">" width="50" height="50" align-items="center"></td>
<td><?php echo $produit["nom_produit"]; ?></td>
<td><?php echo $produit["prix_unitaire"]."€"; ?></td>
<td><?php echo $row["quantite"]; ?></td>
<td><?php echo $prix_total."€"; ?></td>
<td style="text-align: center;"><a href="supprimer_article.php"></a></td>
</tr>
<?php } ?>
<thead>
<tr>
<td bgcolor="white" colspan="4"></td>
<td colspan="1"><strong>Total</strong></td>
<td><strong><?php echo $total."€"; ?></strong></td>
</tr>
</thead>
</tbody>
</table>
<div class="button-container" align="right">
<button class="order" onclick="setTimeout(function(){ window.location.href = 'confirmation.php'; }, 8000);">
    <span class="default">Confirmer Commande</span>
    <span class="success">Commande effectuée

```

```

<svg viewBox="0 0 12 10">
    <polyline points="1.5 6 4.5 9 10.5 1"></polyline>
</svg>
</span>
<div class="box"></div>
<div class="truck">
    <div class="back"></div>
    <div class="front">
        <div class="window"></div>
    </div>
    <div class="light top"></div>
    <div class="light bottom"></div>
</div>
    <div class="lines"></div>
</button>

</div>
<script src='https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js'></script>
<script src=".main.js"></script>
</div>

</body>
</html>

```

### Confirmation.php :

```

<?php
$dsn = 'mysql:host=localhost;dbname=site_e-commerce';
$username = 'root';
$password = "";

require_once "includes/config.php";
require_once "_header.php";

try {
    $db = new PDO($dsn, $username, $password);
} catch (PDOException $e) {
    echo 'Connexion échouée : ' . $e->getMessage();
}

```

```

    exit;
}

$client_id = $_SESSION['id'];

// Mise à jour de l'état de la commande précédente
$update_commande = $db->prepare('UPDATE commande SET etat_commande = 1
WHERE client_id = :client_id AND etat_commande = 0');
$update_commande->execute(array('client_id' => $client_id));

// Création d'une nouvelle commande pour le client actuel
$date_commande = date('Y-m-d H:i:s');
$insert_commande = $db->prepare('INSERT INTO commande (date_commande,
montant_commande, etat_commande, client_id) VALUES (:date_commande, 0,
0, :client_id)');
$insert_commande->execute(array('date_commande' => $date_commande, 'client_id' =>
$client_id));
$num_commande = $db->lastInsertId();

?>
<!DOCTYPE html>
<html>
<head>
    <title>Confirmation de commande</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://fonts.googleapis.com/css?family=Montserrat:400,600,700&display=swap" rel="stylesheet">
<style>
    /* General styles */
    * {
        margin: 0;
        padding: 0;
        box-sizing: border-box;
        font-family: 'Montserrat', sans-serif;
    }
    body {
        background-color: #f2f2f2;
    }
    .container {
        max-width: 800px;

```

```
margin: 0 auto;
padding: 40px;
text-align: center;
position: absolute;
top: 50%;
left: 50%;
transform: translate(-50%, -50%);
}

/* Check mark styles */
.checkmark {
    display: block;
    width: 200px;
    height: 200px;
    margin: 0 auto;
    position: relative;
    transform: translateY(-30px);
}
.checkmark__circle {
    stroke-dasharray: 166;
    stroke-dashoffset: 166;
    stroke-width: 5;
    stroke-miterlimit: 10;
    stroke: #4bb71b;
    fill: none;
    animation: stroke 0.6s cubic-bezier(0.65, 0, 0.45, 1) forwards;
}
.checkmark__check {
    transform-origin: 50% 50%;
    stroke-dasharray: 48;
    stroke-dashoffset: 48;
    stroke-width: 5;
    stroke-miterlimit: 10;
    stroke: #4bb71b;
    fill: none;
    animation: stroke 0.3s cubic-bezier(0.65, 0, 0.45, 1) 0.8s forwards,
        fill 0.3s ease-in-out 0.8s forwards;
}
@keyframes stroke {
    100% {
        stroke-dashoffset: 0;
    }
}
```

```
        }
    }
@keyframes fill {
    100% {
        box-shadow: inset 0px 0px 0px 30px #4bb71b;
    }
}

/* Button styles */
.btn {
    display: inline-block;
    background-color: #4bb71b;
    color: #fff;
    font-size: 18px;
    font-weight: 600;
    padding: 10px 20px;
    border-radius: 5px;
    text-decoration: none;
    transition: background-color 0.3s ease-in-out;
    margin-top: 40px;
}
.btn:hover {
    background-color: #3b9014;
}

/* Additional styles */
h1 {
    font-size: 36px;
    font-weight: 700;
    margin-bottom: 20px;
}
p {
    font-size: 18px;
    font-weight: 400;
    margin-bottom: 30px;
    line-height: 1.5;
}
.icon-container {
    display: flex;
    justify-content: center;
    align-items: center;
```

```

        margin-bottom: 30px;
    }
.icon-container i {
    font-size: 80px;
    color: #4bb71b;
}
</style>
</head>
<body>
    <div class="container">
<div class="icon-container">
<svg class="checkmark" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 60 60">
<circle class="checkmark__circle" cx="30" cy="30" r="24" fill="none"/>
<path class="checkmark__check" fill="none" d="M22 30.5L27.5 36 38 38 25"/>
</svg>
</div>
<h1>Votre commande a été confirmée</h1>
<p>Merci pour votre achat. Nous avons bien reçu votre commande et nous la traitons dès maintenant. Un e-mail de confirmation vous sera envoyé très prochainement.</p>
<a href="index.php" class="btn">Retour à la page d'accueil</a>
</div>

</body>
</html>

```

### Add.php :

```

<?php
$dsn = 'mysql:host=localhost;dbname=site_e-commerce';
$username = 'root';
$password = "";

require_once "includes/config.php";
require_once "_header.php";

try {
    $pdo = new PDO($dsn, $username, $password);
} catch (PDOException $e) {
    echo 'Connexion échouée : ' . $e->getMessage();
}

```

```

if (!isset($_SESSION['username'])) {
    header('Location: login.php');
    exit();
}

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $product_id = $_POST['id'];
    $quantity = $_POST['quantite'];

    // Vérifie si l'utilisateur a déjà une commande en cours
    $users_id = $_SESSION['id'];
    $order_query = $pdo->prepare("SELECT * FROM commande WHERE client_id = ? AND etat_commande = 0");
    $order_query->execute([$users_id]);
    $order = $order_query->fetch(PDO::FETCH_ASSOC);

    if (!$order) {
        // Si l'utilisateur n'a pas de commande en cours, en créer une
        $now = date('Y-m-d H:i:s');
        $order_query = $pdo->prepare("INSERT INTO commande (date_commande, montant_commande, etat_commande, client_id) VALUES (?, ?, ?, ?)");
        $order_query->execute([$now, 0, 1, $users_id]);

        $order_id = $pdo->lastInsertId();
    } else {
        $order_id = $order['num_commande'];
    }

    // Vérifie si le produit est déjà dans le panier
    $cart_item_query = $pdo->prepare("SELECT * FROM article WHERE produit_id = ? AND num_commande = ? AND client_id = ?");
    $cart_item_query->execute([$product_id, $order_id, $users_id]);
    $cart_item = $cart_item_query->fetch(PDO::FETCH_ASSOC);

    if ($cart_item) {
        // Si le produit est déjà dans le panier, mettre à jour la quantité
        $new_quantity = $cart_item['quantite'] + $quantity;
        $total_price = $new_quantity * $cart_item['prix_unitaire'];

        $cart_item_update_query = $pdo->prepare("UPDATE article SET quantite = ?,

```

```

prix_total = ? WHERE id = ?");
$cart_item_update_query->execute([$new_quantity, $total_price, $cart_item['id']]);
} else {
    // Sinon, ajouter le produit au panier
    $product_query = $pdo->prepare("SELECT * FROM produit WHERE id = ?");
    $product_query->execute([$product_id]);
    $product = $product_query->fetch(PDO::FETCH_ASSOC);

    $total_price = $quantity * $product['prix_unitaire'];

    $cart_item_insert_query = $pdo->prepare("INSERT INTO article (quantite, prix_total,
produit_id, num_commande, client_id, etat_valable) VALUES (?, ?, ?, ?, ?, 1)");
    $cart_item_insert_query->execute([$quantity, $total_price, $product_id, $order_id,
$users_id]);
}

// Rediriger l'utilisateur vers la page du panier
header('Location: panier.php');
exit();
} else {
    // Rediriger l'utilisateur vers la page d'accueil si la méthode HTTP n'est pas POST
    header('Location: index.php');
    exit();
}
?>

```

## Étudiant 2: ANNEXE

printer.py

```
import win32print
from zebra import Zebra

class Printer:
    def __init__(self):
        self.z = Zebra('ZDesigner GK420d')

    def send_command(self, barcode):
        if barcode:
            # ZPL command
            label = f"""
                ^XA
                ^FO170,50,^BY3
                ^BEN,110,Y,N
                ^FD{barcode}^FS
                ^XZ
                """
            self.z.output(label)
```

scale.py

```
import serial

class Scale:
    weightProduct = None
    scaleOutput = None

    def __init__(self):
        self.ser = serial.Serial()
        self.ser.port = "COM1"
        self.ser.baudrate = 9600
        self.ser.bytesize = serial.EIGHTBITS
        self.ser.parity = serial.PARITY_NONE
        self.ser.stopbits = serial.STOPBITS_ONE
        self.ser.timeout = 1

    def setWeight(self):
```

```

    self.ser.open()
    self.ser.write(b"s")

    data = self.ser.readline().decode('ascii')

    if data:
        if data.strip().find("M") != -1: # == False
            self.scaleOutput = data.strip().replace("M      ","")
            self.weightProduct = int(self.scaleOutput.replace(" g",""))
        else:
            self.scaleOutput = data.strip().replace("      ","")
            self.weightProduct = int(self.scaleOutput.strip().replace(" g",""))
    else:
        self.weightProduct = None

    self.ser.close()

def getWeight(self):
    return self.weightProduct

def getScaleOutput(self):
    return self.scaleOutput

```

## barcode\_weighing\_scale.py

```

"""
POS Solution System GUI
Author: Githendra Perera
Version: 4.8
"""

from tkinter import *
from io import BytesIO
from PIL import Image, ImageTk
from CTkMessagebox import CTkMessagebox
import customtkinter as ctk
import mysql.connector, requests, time, subprocess, sys

from printer import Printer
from scale import Scale

BUTTONS_IN_A_ROW = 3
IMG_SIZE = (50, 50)

class App(ctk.CTk):

```

```

appWidth, appHeight = 460, 215
rowX, columnY = 2, 0
numberOfProducts = 0
dbConnectionState = False
printer = Printer()
scale = Scale()

def __init__(self):
    super().__init__()
    self.idProduct = None

    self.spawn_x = (self.winfo_screenwidth() - self.appWidth)/2
    self.spawn_y = (self.winfo_screenheight() - self.appWidth)/2

    self.lastButtonNumber = 4
    self.fade = 100

    self.transparent_color = self._apply_appearance_mode(self._fg_color)
    self.attributes("-transparentcolor", self.transparent_color)

    self.config(background = self.transparent_color)

try:
    self.mydb = mysql.connector.connect(
        host = "localhost",
        user = "root",
        password = "",
        database = "site_e-commerce"
    )
    self.mycursor = self.mydb.cursor()
    self.dbConnectionState = True

except mysql.connector.Error as e:
    print("Error reading data from MySQL table", e)
    self.dbConnectionState = False
    self.overrideredirect(1)
    CTkMessagebox(title="Erreur", message="Échec connexion BDD", icon="cancel")

if self.dbConnectionState is True:
    self.attributes("-topmost", True)

    self.frame_top = ctk.CTkFrame(self, corner_radius = 10, width = 10,
border_width=1)
    self.frame_top.grid(sticky="nswe")

    self.frame_top.bind("<B1-Motion>", self.move_window)
    self.frame_top.bind("<ButtonPress-1>", self.oldxyset)

```

```

        self.frame_top.bind("<Map>", self.on_configure)

        self.button_close = ctk.CTkButton(self.frame_top, corner_radius=10, width=10,
height=10, hover=False,
                text="", fg_color="red", command=self.button_event)
        self.button_close.configure(cursor="arrow")
        self.button_close.grid(row=0, column=3, sticky="ne", padx=(5, 10), pady=10)
        self.button_close.bind("<Enter>", lambda e, button = self.button_close: self-
.enter_closeButton(button))
        self.button_close.bind("<Leave>", lambda e, button = self.button_close: self-
.leave_closeButton(button))

        self.button_minimize = ctk.CTkButton(self.frame_top, corner_radius=10,
width=10, height=10, hover=False,
                text="", fg_color="orange", command=self.minimize_window)
        self.button_minimize.configure(cursor="arrow")
        self.button_minimize.grid(row=0, column=2, sticky="ne", padx=5, pady=10)

        self.geometry(f'{self.spawn_x}x{self.spawn_y}')

# it contains the name of each buttons and its colour
self.buttonsArr = [[] for _ in range(2)]
# it contains the name of the products and their img link
self.productsArr = [[] for _ in range(2)]

# get the id of the first perishable product from the produit table
self.mycursor.execute("SELECT MIN(id) AS first_id FROM produit")
firstID = int(str(self.mycursor.fetchone()).replace("(", "").replace(")", ","))
"""))

# get the id of the last perishable product from the produit table
self.mycursor.execute("SELECT MAX(id) AS last_id FROM produit;")
lastID = int(str(self.mycursor.fetchone()).replace("(", "").replace(")", ","))
"""))

for i in range(firstID, lastID + 1):
    # get the name of the products stored in the database and add it in an ar-
ray
    self.mycursor.execute(f"SELECT nom_produit FROM produit WHERE id = {i} AND
TYPE = 'périsable'")
    nameItem = (str(self.mycursor.fetchone())).replace("(", "").replace(")", ","))
    if nameItem != "None":
        self.productsArr[0].append(nameItem)
        self.numberofProducts += 1

    # get the url of the picture stored in the database and add it in an array

```

```

        self.mycursor.execute(f"SELECT URL FROM produit WHERE id = {i} AND TYPE = 'périsable'")
        url = (str(self.mycursor.fetchone())).replace("('", "").replace(")', '", ")
        if url != "None":
            self.productsArr[1].append(url)

    ctk.set_appearance_mode("Dark")
    ctk.set_default_color_theme("green")

    self.title("EAN-13 Code Barre")

    # Weight Label
    weightLabel = ctk.CTkLabel(self.frame_top,
                               text = "Poids")
    weightLabel.grid(row = 0, column = 0,
                     padx = (50, 20), pady = (20, 10),
                     sticky = "ew")

    # Weight Entry Field
    self.weightEntry = ctk.CTkEntry(self.frame_top,
                                     state = 'disabled')
    self.weightEntry.grid(row = 0, column = 1,
                          columnspan = 1, padx = (20, 20),
                          pady = (20, 10), sticky = "ew")

    # Display the weight Button
    weightButton = ctk.CTkButton(self.frame_top, text = "Afficher",
                                 command = self.insertWeight)
    weightButton.grid(row = 1, column = 1,
                      columnspan = 1,
                      padx = (20, 20), pady = (0, 20),
                      sticky = "ew")

    # ----- custom product buttons
-----
    for i in range(self.numberofProducts):

        # buttons and resize images
        img_url = self.productsArr[1][i]
        img = self.load_image_from_url(img_url, IMG_SIZE)

        self.button = ctk.CTkButton(self.frame_top,
                                    image = img, text = self.productsArr[0][i],
                                    compound = "top", command = lambda productName = self.productsArr[0][i]: self.get_product_id(productName))

        # detect the mouse hovering the buttons

```

```

        self.button.bind("<ButtonPress-1>" , lambda e, buttons = self.buttonsArr[0], button = self.button: self.on_click(button, buttons))
        self.button.bind("<Enter>" , lambda e, buttons = self.buttonsArr[0], button = self.button: self.buttonEnterHover(button, buttons))
        self.button.bind("<Leave>" , lambda e, buttons = self.buttonsArr[0], button = self.button: self.buttonLeaveHover(button, buttons))
        self.button.bind("<ButtonRelease-1>" , lambda e, buttons = self.buttonsArr[0], button = self.button: self.buttonClicked(button, buttons))

    if i == 0:
        self.appHeight += 100
    else:
        if i % BUTTONS_IN_A_ROW == 0:
            self.rowX += 1
            self.columnY = 0
            self.appHeight += 100

    if i == 0:
        self.button.grid(column = self.columnY, row = self.rowX,
                          columnspan = 1, padx = (25, 0), pady = 10)
    else:
        self.button.grid(column = self.columnY, row = self.rowX,
                          columnspan = 1, padx = (25 if (i % BUTTONS_IN_A_ROW == 0) else 0, 0),
pady = 10)

    self.buttonsArr[0].append(self.button)
    self.buttonsArr[1].append("green")
    self.columnY += 1

printButton = ctk.CTkButton(self.frame_top,
                           command = self.print_barcode,
                           text = "Imprimez Code Barre")
printButton.grid(row = self.rowX + 1, column = 1,
                 columnspan = 1,
                 padx = (20, 20), pady = (20, 5),
                 sticky = "ew")

quitButton = ctk.CTkButton(self.frame_top, text = "Quitter",
                           command = self.button_event)
quitButton.grid(row = self.rowX + 2, column = 1,
                 columnspan = 1,
                 padx = (20, 20), pady = 5,
                 sticky = "ew")

reloadButton = ctk.CTkButton(self.frame_top,
                            image = self.load_image_from_url("https://cdn-icons-png.flaticon.com/512/560/560450.png", (20, 20)),

```

```

        text = None,
        width = 10, height = 10,
        command = self.reload)
reloadButton.grid(row = self.rowX + 2, column = 2,
                  padx = (20, 20), pady = 5)
#
-----
-----

        self.resizable(False, False)
self.frame_top.configure(width = self.appWidth, height = self.appHeight)

        self.iconphoto(False,
                      self.load_image_from_url("https://cdn-icons-png.flaticon.com/
512/2432/2432797.png", IMG_SIZE))
self.idProduct = None

# ----- ToolBar buttons -----
def enter_closeButton(self, button):
    button.configure(fg_color = "#c42b1c")

def leave_closeButton(self, button):
    button.configure(fg_color = "red")
# -----


def oldxyset(self, event):
    self.oldx = event.x
    self.oldy = event.y

def on_configure(self, event):
    if self.wm_state() != "zoomed":
        self.overrideredirect(1)

def minimize_window(self):
    self.overrideredirect(0)
    self.iconify()

def move_window(self, event):
    self.overrideredirect(1)
    self.y = event.y_root - self.oldy
    self.x = event.x_root - self.oldx
    self.geometry(f'{self.x}+{self.y}')

def button_event(self, event=None):
    if self.fade:
        self.fade_out()
        self.grab_release()

```

```

self.mydb.close()
del scale
del printer
self.destroy()
self.event = event

def fade_out(self):
    for i in range(100, 0, -10):
        if not self.winfo_exists():
            break
        self.attributes("-alpha", i/100)
        self.update()
        time.sleep(1/self.fade)

# ----- Custom Button -----

def buttonLeaveHover(self, button, buttons):
    for b in buttons:
        checkButtonName = str(b)

        if b == button:
            if self.buttonsArr[1][0] if (checkButtonName == ".!ctkframe.!ctkbutton")
else int(checkButtonName.replace(".!ctkframe.!ctkbutton", "")) - self.lastButton-
Number] == "blue":
                b.configure(fg_color = "#1f6aa5") # blue

def buttonClicked(self, button, buttons):
    for b in buttons:
        checkButtonName = str(b)

        if b == button:
            if self.buttonsArr[1][0] if (checkButtonName == ".!ctkframe.!ctkbutton")
else int(checkButtonName.replace(".!ctkframe.!ctkbutton", "")) - self.lastButton-
Number] == "blue":
                b.configure(fg_color = "#144870") # dark blue

            # Simulate a button press event to trigger the button's default behavior
            b.event_generate("<Button-1>")

        else:
            if self.buttonsArr[1][0] if (checkButtonName == ".!ctkframe.!ctkbutton")
else int(checkButtonName.replace(".!ctkframe.!ctkbutton", "")) - self.lastButton-
Number] == "blue":
                b.configure(fg_color = "#1f6aa5") # blue

def buttonEnterHover(self, button, buttons):
    for b in buttons:

```

```

checkButtonName = str(b)

    if b == button:
        if self.buttonsArr[1][0] if (checkButtonName == ".!ctkframe.!ctkbutton")
else int(checkButtonName.replace(".!ctkframe.!ctkbutton", "")) - self.lastButton-
Number] == "blue":
            b.configure(fg_color = "#144870") # dark blue
        else:
            if self.buttonsArr[1][0] if (checkButtonName == ".!ctkframe.!ctkbutton")
else int(checkButtonName.replace(".!ctkframe.!ctkbutton", "")) - self.lastButton-
Number] == "blue":
                b.configure(fg_color = "#1f6aa5") # blue

def on_click(self, button, buttons):
    for b in buttons:
        checkButtonName = str(b)

        if b == button:
            b.configure(fg_color = "#1f6aa5") # blue
            self.buttonsArr[1][0] if (checkButtonName == ".!ctkframe.!ctkbutton") else
int(checkButtonName.replace(".!ctkframe.!ctkbutton", "")) - self.lastButtonNumber]
= "blue"
            id(self.productsArr[0][0] if (checkButtonName == ".!ctkframe.!ctkbutton")
else int(checkButtonName.replace(".!ctkframe.!ctkbutton", "")) - self.lastButton-
Number])

        # Simulate a button press event to trigger the button's default behavior
        b.event_generate("<Button-1>")

    else:
        b.configure(fg_color="#2fa572") # green
        self.buttonsArr[1][0] if (checkButtonName == ".!ctkframe.!ctkbutton") else
int(checkButtonName.replace(".!ctkframe.!ctkbutton", "")) - self.lastButtonNumber]
= "green"

# -----


def insertWeight(self):
    self.scale.setWeight()
    text = self.scale.getScaleOutput()

    if text:
        self.weightEntry.configure(state = 'normal', justify = CENTER)
        self.weightEntry.delete(0, "end")
        self.weightEntry.insert(1, text)
        self.weightEntry.configure(state = 'disabled')
    else:

```

```

        self.weightEntry.configure(state = 'normal', justify = CENTER)
        self.weightEntry.delete(0, "end")
        self.weightEntry.insert(1, "Erreur")
        self.weightEntry.configure(state = 'disabled')

def load_image_from_url(self, url, new_size):
    response = requests.get(url)
    image = Image.open(BytesIO(response.content))
    image = image.resize(new_size)
    return ImageTk.PhotoImage(image)

def generate_barcode(self, product_id, product_weight):
    if product_id is None:
        CTkMessagebox(message = "Choisissez un produit",
                      icon = "warning", option_1 = "Ok")
        return None

    elif product_weight is None:
        CTkMessagebox(message = "Récupérez le poids du produit",
                      icon = "warning", option_1 = "Ok")
        return None

    barcode = int("{}{:06d}{:05d}".format(product_id, product_weight))
    return barcode

def get_product_id(self, product_name):
    if product_name is not None:
        sql_query = "SELECT id FROM produit WHERE nom_produit = %s"
        self.mycursor.execute(sql_query, (product_name,))

        id = int(str(self.mycursor.fetchone()).replace("(", "").replace(")", "").replace(",", ""))
        self.idProduct = id

    def print_barcode(self):
        self.printer.send_command(self.generate_barcode(self.idProduct, self.scale.getWeight()))

    def reload(self):
        self.button_event()
        python = sys.executable
        subprocess.call([python, __file__])
        sys.exit()

    if __name__ == "__main__":
        app = App()
        app.mainloop()

```

## Site Web :

index.php

```
<?php
    require_once "process_orders.php";
?>

<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>Commande Client</title>
        <link rel="stylesheet" href="style.css">
        <script src="updateNonManquant_function.js"></script>
    </head>

    <body>
        <?php
        if ($number_of_orders == 0)
        {
            ?>

            <div class="centerPage">
                <h1>Aucun panier à traiter</h1>
            </div>

            <?php
        }
        else
        {
            ?>
            <h1 class="centerMargin">Traitement des paniers</h1>
            <?php

$checkbox_id = 0;
for ($i = 0; $i < $number_of_orders; $i++)
{
    ?>
    <table>
        <tr>
            <th colspan="3"><?php echo $customerOrder[$i]->getName(); ?></th>
        </tr>
        <tr>
            <th>Produit</th>
            <th>Quantité</th>
            <th><?php echo $customerOrder[$i]->getOrder_date(); ?></th>
        </tr>
```

```

<?php

for ($j = 0; $j < $customerOrder[$i]->getTotalProducts(); $j++)
{
    $product_name = $customerOrder[$i]->getProductName($j);
    $product_quantity = $customerOrder[$i]->getProductQuantity($j);
    $product_id = $customerOrder[$i]->getProductId($j);
    $order_id = $customerOrder[$i]->getOrder_id();
    ?>
    <tr>
        <td><?php echo $product_name; ?></td>
        <td><?php echo $product_quantity; ?></td>
        <td><input type="checkbox" name=<?php echo $product_id ;?>" class="checkbox" data-order-id=<?php echo $order_id; ?>" data-product-quantity=<?php echo $product_quantity; ?>" id=<?php echo $checkbox_id ?>" <?php if($customerOrder[$i]->getProductAvailable($j) == "true") { echo "checked"; } ?>></td>
    </tr>
    <?php
        $checkbox_id++;
    ?
    ?>
</table>
<table style="border-color: transparent;">
    <tr>
        <th colspan="3" style="background-color : transparent;">
            <button id=<?php echo $customerOrder[$i]->getOrder_id(); ?>" onclick="updateNonManquant(<?php echo $customerOrder[$i]->getOrder_id(); ?>, <?php echo $customerOrder[$i]->getTotalProducts();?>)\">Valider</button>
        </th>
    </tr>
</table>

<br> <br>
<?php
}
?
?>
</body>
</html>

```

## process.orders.php

```
<?php
require_once "customerOrder.class.php";

$servername = "mysql-projet-e-commerce.alwaysdata.net";
$username = "312817";
$password = "LucasRatonLaveur";
$dbname = "projet-e-commerce_bdd";

$link = new mysqli($servername, $username, $password, $dbname);
if ($link->connect_error)
{
    error_reporting("Erreur avec les client_id" . $link->connect_error);
}
else
{
    $result = $link->query("SELECT COUNT(num_commande) FROM commande WHERE etat_com-
mande = 1");
    if ($result === false)
    {
        error_reporting("Erreur avec les client_id: " . mysqli_error($link));
    }
    else
    {
        $number_of_orders = $result->fetch_assoc()["COUNT(num_commande)"];

        if ($number_of_orders > 0)
        {
            $order_id = array();
            $result = $link->query("SELECT num_commande FROM commande WHERE etat_commande
= 1 ORDER BY date_commande ASC");
            while($row = $result->fetch_assoc())
            {
                $order_id[] = $row["num_commande"];
            }

            $customerOrder = array();
            for ($i = 0; $i < $number_of_orders; $i++)
            {
                $customerOrder[$i] = new CustomerOrder();
                $customerOrder[$i]->setOrder_id($order_id[$i]);

                $get_client_id = $link->query("SELECT client_id FROM commande WHERE
num_commande = {$customerOrder[$i]->getOrder_id()}");
                $customerOrder[$i]->setID($get_client_id->fetch_assoc()["client_id"]);

                $get_client_id = $link->query("SELECT client_id FROM commande WHERE
num_commande = {$customerOrder[$i]->getOrder_id()}");
            }
        }
    }
}
```

```

num_commande = "{$customerOrder[$i]->getOrder_id()}");

        $get_date_order = $link->query("SELECT date_commande FROM commande WHERE
num_commande = "{$customerOrder[$i]->getOrder_id()}");
        $customerOrder[$i]->setOrder_date($get_date_order->fetch_assoc()["date_com-
mande"]);

        $get_username = $link->query("SELECT username FROM client WHERE id =
'{$customerOrder[$i]->getOrder_id()}'");
        $customerOrder[$i]->setName($get_username->fetch_assoc()["username"]);

        $get_number_of_products = $link->query("SELECT COUNT(nom_produit) FROM pro-
duit WHERE id IN (SELECT produit_id FROM article WHERE num_commande = {$cus-
tomerOrder[$i]->getOrder_id()})");
        $customerOrder[$i]->setTotalProducts($get_number_of_products->fetch_assoc()
["COUNT(nom_produit)"]);

        $get_products_name = $link->query("SELECT nom_produit FROM produit WHERE id
IN (SELECT produit_id FROM article WHERE num_commande = {$customerOrder[$i]-
>getOrder_id()})");

        $count = 0;
        while ($row = $get_products_name->fetch_assoc())
        {
            $customerOrder[$i]->setProductName($count, $row["nom_produit"]);
            $count++;
        }

        for($j = 0; $j < $customerOrder[$i]->getTotalProducts(); $j++)
        {
            $get_product_quantity = $link->query("SELECT quantite FROM article WHERE
num_commande = {$customerOrder[$i]->getOrder_id()} AND produit_id IN (SELECT id
FROM produit WHERE nom_produit = '{$customerOrder[$i]->getProductName($j)}')");
            $customerOrder[$i]->setProductQuantity($j, $get_product_quantity-
>fetch_assoc()["quantite"]);

            $get_product_id = $link->query("SELECT id FROM article WHERE num_commande
= {$customerOrder[$i]->getOrder_id()} AND produit_id IN (SELECT id FROM produit
WHERE nom_produit = '{$customerOrder[$i]->getProductName($j)}')");
            $customerOrder[$i]->setProductId($j, $get_product_id->fetch_assoc()
["id"]);

            $result = $link->query("SELECT IF(etat_stock = 1, 'true', 'false') AS
non_manquant_checked FROM article WHERE id = {$customerOrder[$i]->getProduc-
tId($j)}");
            $customerOrder[$i]->setProductAvailable($j, $result->fetch_assoc()
["non_manquant_checked"]);

```

```
        }
    }
}
}

mysqli_close($link);
?>
```

### customerOrder.class.php

```
<?php
require_once "product.class.php";

class CustomerOrder
{
    private $name;
    private $product_catalog = array();
    private $total_products;
    private $order_date;
    private $id;
    private $order_id;

    public function setTotalProducts($number)
    {
        $this->total_products = $number;
        for($i = 0; $i < $this->total_products; $i++)
        {
            $this->product_catalog[$i] = new Product;
        }
    }

    public function setID($id)
    {
        $this->id = $id;
    }

    public function setOrder_id($id)
    {
        $this->order_id = $id;
    }

    public function setName($name)
    {
        $this->name = $name;
    }
}
```

```
public function setProductName($number, $item)
{
    $this->product_catalog[$number]->setProductName($item);
}

public function setProductQuantity($number, $quantity)
{
    $this->product_catalog[$number]->setProductQuantity($quantity);
}

public function setProductId($number, $product_id)
{
    $this->product_catalog[$number]->setProductId($product_id);
}

public function setProductAvailable($number, $id)
{
    $this->product_catalog[$number]->setProductAvailability($id);
}

public function setOrder_date($datetime)
{
    $this->order_date = $datetime;
}

public function getID()
{
    return $this->id;
}

public function getOrder_id()
{
    return $this->order_id;
}

public function getName()
{
    return $this->name;
}

public function getTotalProducts()
{
    return $this->total_products;
}

public function getProductName($item_number)
{
    return $this->product_catalog[$item_number]->getProductName();
```

```

}

public function getProductQuantity($item_number)
{
    return $this->product_catalog[$item_number]->getProductQuantity();
}

public function getProductId($item_number)
{
    return $this->product_catalog[$item_number]->getProductId();
}

public function getProductAvailable($item_number)
{
    return $this->product_catalog[$item_number]->getProductAvailability();
}

public function getOrder_date()
{
    return $this->order_date;
}
?>

```

### product.class.php

```

<?php
class Product
{
    private $name;
    private $quantity;
    private $id;
    private $availability;

    public function setProductName($name)
    {
        $this->name = $name;
    }

    public function setProductQuantity($quantity)
    {
        $this->quantity = $quantity;
    }

    public function setProductId($id)
    {

```

```

    $this->id = $id;
}

public function setProductAvailability($availability)
{
    $this->availability = $availability;
}

public function getProductName()
{
    return $this->name;
}

public function getProductQuantity()
{
    return $this->quantity;
}

public function getProductId()
{
    return $this->id;
}

public function getProductAvailability()
{
    return $this->availability;
}
}

?>
```

### style.css

```

html {
    font-family: sans-serif;
}

table {
    border-collapse: collapse;
    border: 2px solid rgb(200,200,200);
    letter-spacing: 1px;
    font-size: 0.8rem;
    table-layout: fixed;
    width: 400px;
    margin-left: auto;
    margin-right: auto;
}
```

```
.centerPage {
height: 100vh;
display: flex;
flex-direction: column;
justify-content: center;
text-align: center;
}

.centerMargin {
text-align: center;
}

td, th {
border: 1px solid rgb(190,190,190);
padding: 10px 20px;
text-align: center;
}

th {
background-color: rgb(235,235,235);
}

tr:nth-child(even) td {
background-color: rgb(250,250,250);
}

tr:nth-child(odd) td {
background-color: rgb(245,245,245);
}

/* ----- checkbox ----- */

[type=checkbox]{
width: 1rem;
height: 1rem;
color: dodgerblue;
vertical-align: middle;
-webkit-appearance: none;
background: none;
border: 0;
outline: 0;
flex-grow: 0;
border-radius: 50%;
background-color: #FFFFFF;
transition: background 300ms;
cursor: pointer;
```

```
}

/* Pseudo element for check styling */

[type=checkbox]::before {
  content: "";
  color: transparent;
  display: block;
  width: inherit;
  height: inherit;
  border-radius: inherit;
  border: 0;
  background-color: transparent;
  background-size: contain;
  box-shadow: inset 0 0 0 1px #CCD3D8;
}

/* Checked */

[type=checkbox]:checked {
  background-color: currentcolor;
}

[type=checkbox]:checked::before {
  box-shadow: none;
  background-image: url("data:image/svg+xml,%3Csvg xmlns='http://www.w3.org/2000/svg' width='24' height='24' viewBox='0 0 24 24'%3E %3Cpath d='M15.88 8.29L10 14.17l-1.88-1.88a.996.996 0 1 0-1.41 1.41l2.59c.39.39 1.02.39 1.41 0L17.3 9.7a.996.996 0 0 0 0-1.41c-.39-.39-1.03-.39-1.42 0z' fill='%23fff'%3E %3C/svg%3E");
}

/* Disabled */

[type=checkbox]:disabled {
  background-color: #CCD3D8;
  opacity: 0.84;
  cursor: not-allowed;
}

/* IE */

[type=checkbox]::-ms-check {
```

```
content: "";
color: transparent;
display: block;
width: inherit;
height: inherit;
border-radius: inherit;
border: 0;
background-color: transparent;
background-size: contain;
box-shadow: inset 0 0 0 1px #CCD3D8;
}

[type=checkbox]:checked::-ms-check {
  box-shadow: none;
  background-image: url("data:image/svg+xml,%3Csvg xmlns='http://www.w3.org/2000/svg' width='24' height='24' viewBox='0 0 24 24'%3E %3Cpath d='M15.88 8.29L10 14.17l-1.88-1.88a.996.996 0 1 0-1.41 1.41L2.59 2.59c.39.39 1.02.39 1.41 0L17.3 9.7a.996.996 0 0 0 0-1.41c-.39-.39-1.03-.39-1.42 0z' fill='%23fff'/%3E %3C/svg%3E");
}

button {
  cursor: pointer;
  outline: 0;
  display: inline-block;
  font-weight: 400;
  line-height: 1.5;
  text-align: center;
  background-color: transparent;
  border: 1px solid transparent;
  padding: 6px 12px;
  font-size: 1rem;
  border-radius: .25rem;
  transition: color .15s ease-in-out,background-color .15s ease-in-out,border-color .15s ease-in-out,box-shadow .15s ease-in-out;
  color: #0d6efd;
  border-color: #0d6efd;
}

button:hover {
  color: #fff;
  background-color: #0d6efd;
  border-color: #0d6efd;
}
```

## updateNonManquant\_function.js

```
function updateNonManquant(order_id, totalProducts)
{
    for (let i = 0; i < totalProducts; i++)
    {
        var checkbox = document.getElementById(i);
        var product_id = checkbox.name;
        var button = checkbox.getAttribute("data-order-id");
        var product_quantity = checkbox.getAttribute("data-product-quantity");

        if (button.includes(order_id))
        {
            var date = new Date().toISOString().slice(0, 19).replace('T', ' ');
            var xhr = makeRequest();
            var sql;

            if (checkbox.checked)
            {
                sql = `UPDATE article SET etat_stock = 1 WHERE id = ${product_id} AND
num_commande = ${order_id}`;
                xhr.send(JSON.stringify({sql: sql}));

                xhr = makeRequest();

                sql = `UPDATE produit SET stock = stock - ${product_quantity} WHERE id = ${
product_id}`;
                xhr.send(JSON.stringify({sql: sql}));
            }
            else
            {
                sql = `UPDATE article SET etat_stock = 0 WHERE id = ${product_id} AND
num_commande = ${order_id}`;
                xhr.send(JSON.stringify({sql: sql}));
            }

            xhr = makeRequest();

            sql = `UPDATE commande SET etat_commande = 2, date_traitement = '${date}'
WHERE num_commande = ${order_id}`;
            xhr.send(JSON.stringify({sql: sql}));
        }
    }
    window.location.reload();
}

function makeRequest()
{
```

```

var xhr = new XMLHttpRequest();
var url = "db_query.php";

xhr.onreadystatechange = function()
{
  if (this.readyState == 4 && this.status == 200)
  {
    var results = JSON.parse(this.responseText);
  }
};

xhr.open("POST", url, true);
xhr.setRequestHeader("Content-Type", "application/json");

return xhr;
}

```

### db\_query.php

```

<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "site_e-commerce";

$sql = json_decode(file_get_contents("php://input"))->sql;

$link = mysqli_connect($servername, $username, $password, $dbname);
$result = mysqli_query($link, $sql);

mysqli_close($link);
?>

```

# Étudiant 3: ANNEXE

IHM :

MainWindow

**Facture**

De  A

MarketMate Nom du client

MarketMate@email.com client@email.com

3 rue Albert Samain

telephone: 0125262898

numéro de facture :  numero facture et date

Date :

| nom article | cout unitaire | quantite | montant |
|-------------|---------------|----------|---------|
|             |               |          |         |

code barre :  affichage des produits

Montant total :  Calcul Montant  impression

## Code Qt :

### Facturation.h :

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QtSql>
#include <QMainWindow>
#include <iostream>
#include <QtWidgets>
#include <QPrinter>
#include <QFile>
#include <QTextStream>
#include <QFileDialog>
#include <QMessageBox>
#include <QString>
#include <QWidget>
#include <cstdlib>
#include <ctime>
#include <string>
#include <iostream>
#include <QPrintDialog>
#include <QRect>
#include <QScreen>
#include <QDesktopWidget>

using namespace std;

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
    void getCommande();

private slots:
    void on_pushButtonfacture_et_date_clicked();
    void on_pushButton_save_clicked();
    void on_pushButton_produit_clicked();
    void on_pushButton_Calcul_clicked();
    void on_pushButton_imp_clicked();
}
```

```

void on_pushButton_clicked();
void on_pushButton_2_clicked();
void on_comboBox_currentTextChanged(const QString &arg1);

private:
Ui::MainWindow *ui;
QSqlDatabase db;
QString line2;
QString id = "1";
QString fileName = "";

};

#endif // MAINWINDOW_H

```

## Facturation.cpp :

```

#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    db = QSqlDatabase::addDatabase("QODBC");
    // set the database connection parameters
    db.setHostName("localhost");
    db.setDatabaseName("site_e-commerce");
    db.setUserName("root");
    db.setPassword("");
    //    db.setConnectOptions("MySQL ODBC 8.0 unicode Driver");

    // open the database connection
    if (!db.open()) qDebug() << "Failed to connect to database:" << db.lastError().text();
    else qDebug() << "Connexion réussite";

    getCommande();
    on_comboBox_currentTextChanged(id);
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::on_pushButtonfacture_et_date_clicked()

```

```

{
    srand(time(NULL));

    int randomNumber = rand();
    ui->lineEdit_facture->setText(QString::number(randomNumber));
    QDate date = QDate::currentDate();
    ui->lineEdit_date->setText(date.toString(Qt::ISODate));
}

void MainWindow::on_pushButton_save_clicked()
{
    fileName = QFileDialog::getSaveFileName(this, "Save to PDF", "", "*.pdf");
    if (fileName.isEmpty()) return;
    if (!fileName.endsWith(".pdf")) fileName += ".pdf";

    QPrinter printer(QPrinter::PrinterResolution);
    printer.setOutputFormat(QPrinter::PdfFormat);
    printer.setOutputFileName(fileName);

    QPainter painter;
    painter.begin(&printer);
    render(&painter);
    painter.end();
    qDebug() << fileName;
}

/*+ "+ requete.value("NOM_PRODUIT").toString() + "+ requete-
value("PRIX_UNITAIRE").toString() + "+"
requete.value("POIDS").toString()
+ "+ requete.value("STOCK").toString() + "+requete-
value("TYPE").toString();*/
}

void MainWindow::on_pushButton_produit_clicked()
{
    QString codebarre = ui->textEdit_CodeBarre->toPlainText();
    QStringList codebarreList;
    int startIndex = 0;
    int chunkSize = 6;

    while (startIndex < codebarre.length()) {
        QString chunk = codebarre.mid(startIndex, chunkSize);
        codebarreList.append(chunk);
        startIndex += chunkSize;
    }

    /*for (int i = 0; i <= 2; i++) {
        while(i < codebarreList.length() && codebarreList[i]=='0') {
            i++;
        }
        codebarreList.remove(0,i);
    }*/
}

codebarreList[0] = codebarreList[0].remove(QRegExp("^[0]*"));
codebarreList[1] = codebarreList[1].remove(QRegExp("^[0]*"));
//sert à retirer les 0 devant un chiffre supérieur à 0
codebarreList[1] = codebarreList[1].remove(codebarreList[1].size() - 1,
1); //sert à enlever le dernier chiffre
}

```

```

ui->textEdit_CodeBarre->clear();
qDebug() << codebarreList[0] << endl;
qDebug() << codebarreList[1] << endl;

QSqlQuery requete;
if(requete.exec("SELECT * FROM produit WHERE id = " + codebarreList[0])) {
    qDebug() << "Ok - requete";

    // Boucle qui permet de parcourir les enregistrements renvoyés par la
requete
    while(requete.next()) {
        // On accède ici aux différents champs par leurs noms, il est
également possible.
        // d'y accéder par leur index : requete.value(0)
        qDebug() << requete.value("id") << " " << requete.value("NOM_PRO-
DUIT") << " " << requete.value("PRIX_UNITAIRE") << " "
            << requete.value("POIDS") << " " << requete.value("STOCK")
            << requete.value("TYPE");

        QString id = (requete.value("NOM_PRODUIT")).toString();
        //ui->textEdit->append(id);
        ui->tableWidget_data->insertRow(0);
        ui->tableWidget_data->setItem(0, 0, new QTableWidgetItem(id));

        QString prix = (requete.value("PRIX_UNITAIRE")).toString();
        //ui->textEdit->append(prix);
        ui->tableWidget_data->setItem(0, 1, new QTableWidgetItem(prix + " "
€/kg));

        QString poids = codebarreList[1];
        ui->tableWidget_data->setItem(0, 2, new QTableWidgetItem(poids + " "
g));

        int num = poids.toInt();
        float num2 = num;
        QString montant = QString::number(prix.toInt() * (num2 / 1000));
        ui->tableWidget_data->setItem(0, 3, new QTableWidgetItem(montant +
" €"));

    }
}

void MainWindow::on_pushButton_Calcul_clicked()
{
    // Get the number of rows in the table
    int numRows = ui->tableWidget_data->rowCount();

    // Loop through the rows and calculate the total price
    double totalPrice = 0.0;
    for (int i = 0; i < numRows; i++)
    {
        // Get the price text from column 3 of the current row
        QString priceText = ui->tableWidget_data->item(i, 3)->text();

        // Remove the "euros" sign from the price text
        priceText.remove(" €");
}

```

```

    // Convert the price to a double
    double price = priceText.toDouble();

    // Add the price to the total
    totalPrice += price;

    // Debugging message to show the price being added to the total
    qDebug() << "Price in row " << i << " is " << price << " and total is
now " << totalPrice;
}

// Set the total price in the lineEdit
ui->lineEdit_PrixTotal->setText(QString::number(totalPrice) + " €");
//sert à écrire dans un fichier texte la valeur de la facture
QFile file("test.txt");
if (!file.open(QIODevice::WriteOnly | QIODevice::Text)) {
    cout << "erreur file" << endl;
}

// Écriture dans le fichier
QTextStream out(&file);
out << QString::number(totalPrice) << endl;

// Fermeture du fichier
file.close();

}

void MainWindow::on_pushButton_imp_clicked()
{
    // Créer une instance de QPrinter
    QPrinter printer;
    QFile file(fileName);
    if (!file.open(QIODevice::ReadOnly | QIODevice::Text)) {
        qDebug() << "Failed to open file";
        return;
    }

    // Ouvrir une boîte de dialogue pour choisir l'imprimante et les paramètres
    // d'impression
    QPrintDialog printDialog(&printer, this);
    if (printDialog.exec() == QDialog::Accepted) {
        // Créer un QPainter pour dessiner sur la page imprimée
        QPainter painter(&printer);

        // Dessiner l'interface sur la page
        ui->centralWidget->render(&painter);

        // Terminer l'impression
        painter.end();
    }
    file.close();
}

void MainWindow::on_pushButton_clicked()
{
    getCommande(); //affiche les info client recuperer
}

```

```

}

void MainWindow::on_comboBox_currentTextChanged(const QString &arg1)
{
    id = arg1;//recupere le texte
    QString username = "";
    QString email = "";

    QSqlQuery requete3;
    if(requete3.exec("SELECT * FROM commande WHERE num_commande = " + id)) {
        qDebug() << "Ok - requete";

        // Boucle qui permet de parcourir les enregistrements renvoyés par la
        // requête
        while(requete3.next()) {
            id = (requete3.value("users_id")).toString();
        }
    }
    if(requete3.exec("SELECT * FROM client WHERE id = " + id))
    {
        while(requete3.next())
        {
            username = requete3.value("username").toString();
            email = requete3.value("email").toString();
        }
    }
    ui->lineEdit_username->setText(username);
    ui->lineEdit_email->setText(email);
}

void MainWindow::getCommande()
{
    ui->comboBox->clear();
    QString id = "";
    QSqlQuery requete3;
    if(requete3.exec("SELECT * FROM commande")) {
        qDebug() << "Ok - requete";

        // Boucle qui permet de parcourir les enregistrements renvoyés par la
        // requête
        while(requete3.next()) {
            id = (requete3.value("num_commande")).toString();
            ui->comboBox->addItem(id);
            qDebug() << id;
        }
    }
}

```

Main.cpp :

```
#include "mainwindow.h"
#include < QApplication>
#include <QtWidgets>
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;

    w.show();

    return a.exec();
}
```

### Annexe de la partie paiement

Code pour l'écriture d'une valeur sur la carte ou le badge :

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

#define RST_PIN          5           // Configurable, see typical pin layout above
#define SS_PIN           53          // Configurable, see typical pin layout above

MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance

void setup(){
    Serial.begin(9600); // Initialize serial communications with the PC
    SPI.begin(); // Init SPI bus
    mfrc522.PCD_Init(); // Init MFRC522 card
    Serial.println(F("Write personal data on a MIFARE PICC "));
}

void loop(){
```

```

// Prepare key - all keys are set to FFFFFFFF at chip delivery from the factory.
MFRC522::MIFARE_Key key;
for(byte i = 0; i <6; i++)key.keyByte[i] = 0xFF;

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

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

Serial.print(F("Card UID:"));      //Dump UID
for(byte i = 0; i <mfrc522.uid.size; i++){
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
}
Serial.print(F(" PICC type: "));   // Dump PICC type
MFRC522::PICC_Type piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
Serial.println(mfrc522.PICC_GetTypeName(piccType));

bytebuffer[34];
byte block;
MFRC522::StatusCode status;
byte len;

Serial.setTimeout(20000L); // wait until 20 seconds for input from serial

// Ask for personal data: montant
Serial.println(F("Entrez le montant de la carte, se terminant par #"));
len = Serial.readBytesUntil('#', (char*)buffer, 30); // read montant from serial
for(byte i = len; i <30; i++)buffer[i] = ' ';      // pad with spaces

block = 1;
status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, &key,
&(mfrc522.uid));
if(status != MFRC522::STATUS_OK){
    Serial.print(F("PCD_Authenticate() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
} else Serial.println(F("PCD_Authenticate() success: "));

// Write block

```

```

status = mfrc522.MIFARE_Write(block, buffer, 16);
if(status != MFRC522::STATUS_OK){
    Serial.print(F("MIFARE_Write() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
} else Serial.println(F("MIFARE_Write() success: "));

delay(1000); // wait a second between loops
}

```

Code pour la lecture de la valeur écrite sur la carte :

```

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

#define RST_PIN      5          // Configurable, see typical pin layout above
#define SS_PIN       53         // Configurable, see typical pin layout above

MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance

// ****
*****//****

void setup(){
    Serial.begin(9600); // Initialize serial communications with the PC
    SPI.begin(); // Init SPI bus
    mfrc522.PCD_Init(); // Init MFRC522 card
    Serial.println(F("Read personal data on a MIFARE PICC:")); //shows in serial
that it is ready to read
}

// ****
*****//****

void loop(){

    // Prepare key - all keys are set to FFFFFFFF at chip delivery from the factory.
    MFRC522::MIFARE_Key key;
    for(byte i = 0; i < 6; i++)key.keyByte[i] = 0xFF;
}

```

```

//some variables we need
byte block;
byte len;
MFRC522::StatusCode status;

//-----

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

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

Serial.println(F("**Card Detected:**"));

//-----

mfrc522.PICC_DumpDetailsToSerial(&(mfrc522.uid)); //dump some details about the card

//mfrc522.PICC_DumpToSerial(&(mfrc522.uid));      //uncomment this to see all blocks in hex

//-----

Serial.print(F("Montant: "));

bytebuffer1[18];

block = 4;
len = 18;

//----- GET Montant
status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, 4, &key,
&(mfrc522.uid)); //line 834 of MFRC522.cpp file
if(status != MFRC522::STATUS_OK){
    Serial.print(F("Authentication failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
}

status = mfrc522.MIFARE_Read(block, buffer1, &len);

```

```

if(status != MFRC522::STATUS_OK){
    Serial.print(F("Reading failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
}

//PRINT montant
for(uint8_t i = 0; i <16; i++){
{
    if(buffer1[i] != 32)
    {
        Serial.write(buffer1[i]);
    }
}
Serial.print(" ");
//----- GET montant
bytebuffer2[18];
block = 1;
status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, 1, &key,
&(mfrc522.uid)); //line 834
if(status != MFRC522::STATUS_OK){
    Serial.print(F("Authentication failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
}
status = mfrc522.MIFARE_Read(block, buffer2, &len);
if(status != MFRC522::STATUS_OK){
    Serial.print(F("Reading failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
}
//PRINT Montant
for(uint8_t i = 0; i <16; i++){
    Serial.write(buffer2[i] );
}

//-----
Serial.println(F("\n**End Reading**\n"));

delay(1000); //change value if you want to read cards faster

mfrc522.PICC_HaltA();
mfrc522.PCD_StopCrypto1();
}

```

