

Présentation du projet ALBIZIA

https://albizia-techno.fr

Vous devez vous connecter pour accéder au site !

CONNEXION

Identifiant

Mot de Passe

CONNEXION

Maquette page de connexion

https://albizia-techno.fr

Statistiques ALBIZIA

DECONNEXION

Types (nombre)

Commandes (nombre)

Liste des Types (lien)

Liste des Commandes (lien)

Couleurs (nombre)

Clients (nombre)

Liste des Couleurs (lien)

Liste des Clients (lien)

Commandes à préparer (nb) le : (date)

Nom:	Prénom:	Date	Heure:	Type:	Couleur:	Etat:	Prix Total:
....
....
....

Maquette page d'accueil utilisateur

https://albizia-techno.fr

Statistiques ALBIZIA

Tableau de Bord

Les Types

Les Couleurs

Les Statuts

Les Commandes

Les Clients

Les utilisateurs

Connecté Nom Prénom

Se Déconnecter

Types (nombre)

Commandes (nombre)

Liste des Types (lien)

Liste des Commandes (lien)

Couleurs (nombre)

Clients (nombre)

Liste des Couleurs (lien)

Liste des Clients (lien)

Commandes à préparer (nb) le : (date)

Nom:	Prénom:	Date	Heure:	Type:	Couleur:	Etat:	Prix Total:
....
....
....

Maquette page d'accueil administrateur

1/32

J'ai réalisé ces maquettes à l'aide de l'outil Escalidraw et le cahier des charges élaboré par mon tuteur de stage Déon Alexandre PDG de k-technologies et le client ALBIZIA fleuriste situé à Pouzauges.

L'objet de la demande

- La société Albizia (fleuriste) à fait appelle à la société k-technologies spécialisée dans le développement web afin de lui mètre en place une application lui permettant de gérer les commandes de ses clients de la prise de commande en passant par la préparation jusqu'à la livraison ou l'enlèvement au magasin.
- L'administrateur aura tous les droits ,c'est lui qui devra créer le ou les utilisateurs qui pourront enregistrer les commandes et les clients ainsi que les états d'avancement des commandes.
- L'administrateur pourra :
- Gérer Les Types(variétés de fleurs), Les Couleurs, Les Utilisateurs : ajout, modification, activation, désactivation.
- Gérer Les Clients : ajout, modification, activation, désactivation, suppression.
- Gérer Les Commandes : visualisation, impression,ajout, modification, suppression avec gestion des statuts de chaque commande et rechercher par tri (par date, par client, par type).
- La prise de commande l'utilisateur pourra:

Rechercher un client ,si il est enregistré (vérification des ses cordonnées), si il n'est pas enregistré ,création (nom, prénom,email, tel, adresse postale).Choisir le type et couleur parmi la liste et la saisie du budget client.

Cocher si le client souhaite un ruban et/ou une carte message (avec un champ texte libre).

L'objet de la demande

- ▢ Description de la commande :
- ▢ Saisie de la date et l'heure de Livraison ou Enlèvement (si lieu de livraison différent de l'adresse client enregistrée, alors ajout de l'adresse également enregistrée pour les prochaines commandes), le client pourra avoir plusieurs adresse de livraison liées à son compte
- ▢ Liste déroulante avec un état (payée, non payée, contrat).
- ▢ Pour les contrats il devra entrer un nombre de mois et une date récurrente (exemple : si le client veut être livré pendant 3 mois le 15 de chaque mois).
- ▢ Les commandes doivent être affichées dans l'ordre par date et heure (les commandes à la date du jour doivent apparaître avec un icône différent), en cliquant sur l'icône de la commande on accède à son détail avec un bouton « FIN PREPARATION » qui basculera la commande « A LIVRER » ou « ENLEVEMENT »
- ▢ Les commandes « A LIVRER » auront leur liste par ordre de date et heure ainsi que les commandes « ENLEVEMENT » et toujours repérées par l'icon urgent si identique à la date du jour.
- ▢ Une fois la commande Livrée ou Enlevée au magasin un bouton finaliser changera le statut de celle ci ainsi que l'état à commande payée.

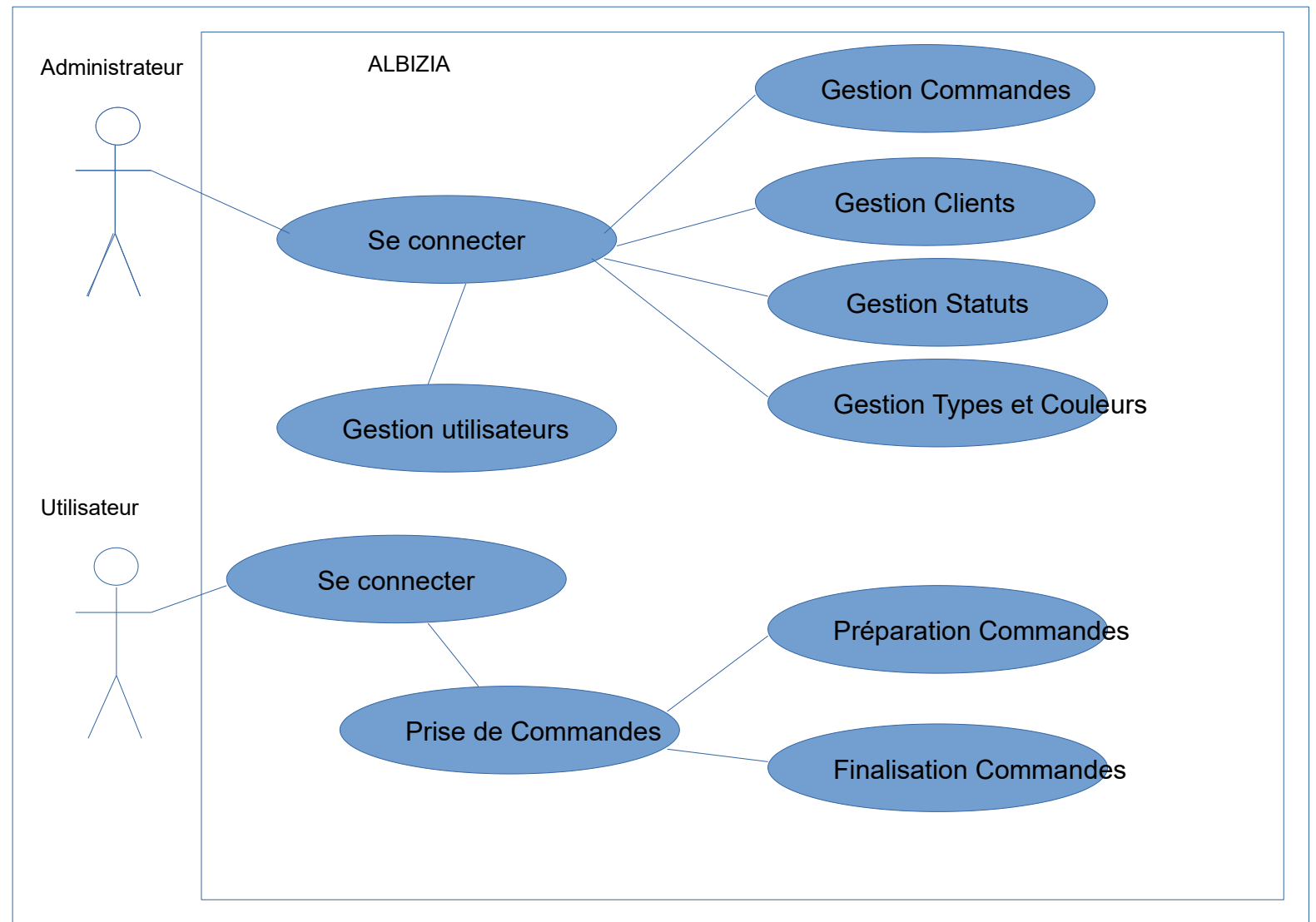
Spécifications fonctionnelles

itération	catégorie	nom	description	démonstration
1	Création des maquettes	escalidraw	Création des maquettes à partir du cahier des charges fourni par mon tuteur	Création des maquettes : page de connexion, page d'accueil utilisateur et administrateur
1	Création de la base de données	mySql	Création de la base de données albizia composée de 8 Tables	Les Tables : Utilisateurs, Clients, Commandes, AdresseLivraisons, Etats, Types, Couleurs, Statuts,
1	Enregistrement d'un utilisateur	Administrateur	En tant qu'administrateur je peux enregistrer des utilisateurs en base données	L'administrateur attribut un login et mot de passe à un utilisateur il est le seul à le faire
1	Se connecter	connexion	En tant qu'utilisateur je peux me connecter au site d'albizia avec mon login et mot de passe	Suite à la saisie correcte du login et mot de passe l'utilisateur accède à la page d'accueil sinon aucune redirection
1	Ajout ou recherche d'un client	utilisateur	L'utilisateur peut rechercher un client existant dans la liste ou ajouter un nouveau client	Avec un champ de saisie l'utilisateur recherche le client qui s'affichera dans la liste déroulante sinon il pourra ajouter ce client
1	Ajout d'une commande	utilisateur	L'utilisateur peut ajouter la commande une fois le client sélectionné et lui attribué une date et heure	l'utilisateur peut choisir entre la date et heure de Livraison ou d'enlèvement au magasin et valider la commande
1	Recherche d'une commande	utilisateur	L'utilisateur peut rechercher commande, voir le détail de la commande	l'utilisateur peut rechercher une commande par date, états, statuts, par nom du client
1	Préparation d'une commande	utilisateur	Quand l'utilisateur prépare une commande il doit la basculé dans le statut correspondant	L'utilisateur une fois la commande préparée il clique sur un bouton qui changera le statut de celle ci
1	Finalisation d'une commande	utilisateur	La finalisation de la commande se fait après la livraison ou l'enlèvement au magasin	L'utilisateur une fois la commande Livrée ou Enlevée au magasin il clique sur un bouton qui changera le statut de celle ci

5/32

Spécifications fonctionnelles

Cas d'utilisation



6/32

Spécifications fonctionnelles

Modèle Vue Relationnel

v	albizia Clients
🔑	id_client : int(11)
📄	nomClient : varchar(50)
📄	prenomClient : varchar(50)
📄	mailClient : varchar(150)
📄	telClient : varchar(20)
📄	adresseClient : varchar(255)
#	cpClient : int(11)
📄	villeClient : varchar(50)
#	activerClient : int(11)

v	albizia Utilisateurs
🔑	id_user : int(11)
📄	nom : varchar(50)
📄	prenom : varchar(50)
📄	mail : varchar(150)
📄	tel : varchar(20)
📄	adresse : varchar(255)
#	cp : int(11)
📄	ville : varchar(50)
📄	mdp : varchar(255)
📄	login : varchar(50)
#	activer : int(11)

v	albizia Commandes
🔑	id_commande : int(11)
📅	dateCommande : date
📅	dateLivraison : datetime
📅	dateEnlevement : datetime
#	quantiteCommande : int(11)
#	prixCommande : decimal(6,2)
#	id_etat : int(11)
#	id_couleur : int(11)
📄	ruban : varchar(250)
#	id_type : int(11)
📄	carteMessage : varchar(250)
#	id_client : int(11)
#	id_statut : int(11)
📅	dateContrat : datetime
#	id_adresseLiv : int(11)

v	albizia AdresseLivraisons
🔑	id_adresseLiv : int(11)
📄	adresseLiv : varchar(250)
#	cpLiv : int(11)
📄	villeLiv : varchar(50)
#	activerAdLiv : int(11)

v	albizia Etats
🔑	id_etat : int(11)
📄	libEtat : varchar(80)
#	activerEtat : int(11)

v	albizia Couleurs
🔑	id_couleur : int(11)
📄	libCouleur : varchar(80)
📄	statutCouleur : varchar(80)
#	activerCouleur : int(11)

v	albizia Types
🔑	id_type : int(11)
📄	libBouquet : varchar(80)
📄	statutBouquet : varchar(80)
#	prixVente : decimal(4,2)
#	quantiteBouquet : int(11)
#	activerType : int(11)

v	albizia Statuts
🔑	id_statut : int(11)
📄	libStatut : varchar(50)
#	actifStatut : int(11)

Spécifications techniques

PS **PhpStorm**



- ▮ Architecture technique
 - ▮ Environnement
 - ▮ Escalidraw
 - ▮ Server FTP
 - ▮ PHP 8.2
 - ▮ IDE : PhpStorm
 - ▮ Framework CSS front : Bootstrap 5
 - ▮ HTML , JavaScript
 - ▮ Documentation PHP , Bootstrap, MDN
 - ▮ SGBDR MySQL

Cas d'utilisation : Accès à la page d'accueil

Développé par Stéphane Bausseron

Présentation de la partie applicative

The mockup shows a web browser window with the URL 'https://albizia-techno.fr'. The page title is 'Statistiques ALBIZIA'. On the left is a 'DECONNEXION' button. The main content area is divided into four sections, each with an icon and a link:

- Types (nombre): Liste des Types (lien)
- Commandes (nombre): Liste des Commandes (lien)
- Couleurs (nombre): Liste des Couleurs (lien)
- Clients (nombre): Liste des Clients (lien)

Below these is a section titled 'Commandes à préparer (nb) le : (date)' with a table of pending orders:

Nom:	Prénom:	Date	Heure:	Type:	Couleur:	Etat:	Prix Total:	
....	icon
....	icon
....	icon

Cas nominal :

Suite à la connexion, l'utilisateur arrive sur une page qui affiche les 4 liens dont l'utilisateur a besoin pour la prise et l'acheminement des commandes ainsi que la liste des commandes avec le statut à préparer.

L'utilisateur voit directement le nombre de commandes à préparer et en cliquant sur l'icône de la commande accède aux détails de celle-ci qui lui permettra d'identifier si c'est une date de livraison ou une date d'enlèvement au magasin. Elles sont triées directement par date et heure de la plus proche à la plus éloignée et celles identiques à la date du jour sont identifiées par un icône urgent.

Cas d'utilisation : Accès à la page d'accueil

Développé par Stéphane Bausseron

Explication technique

```
<div class="col-lg-6 col-sm-5 mt-sm-0 mt-4">
  <div class="card mb-2">
    <div class="card-header p-3 pt-2 bg-transparent">
      <div class="icon icon-lg icon-shape bg-gradient-success shadow-success text-center border-radius-xl mt-n4 position-absolute">
        <i class="material-icons opacity-10">receipt_long</i>
      </div>
      <div class="text-end pt-1">
        <h4 class="text-sm mb-0 text-capitalize">Commandes</h4>
        <h4 class="mb-0"><?php echo $resultCommandes ?></h4>
      </div>
    </div>
    <hr class="horizontal my-0 dark">
    <div class="card-footer p-1">
      <a href="v_listCommande.php"><h4>Liste des Commandes</h4></a>
      <div class="text-end">
        <a href="v_listEnlevement.php"></a>
        <a href="v_listLivrable.php"></a>
      </div>
    </div>
  </div>
</div>
```

Voici ici l'extrait du code HTML de l'affichage de la carte Les Commandes de la page d'accueil qui comporte des classes « bootstrap » qui permettent le « responsive » sur les écrans de type tablette comme demandé par le client, on peut également voir une variable php qui va permettre dynamiquement d'apporter le résultat du nombre de commandes enregistrées en base de données grâce à une requête sql. Cette carte est composée également de 3 ancres qui vont rediriger l'utilisateur soit à la liste de toutes les commandes soit à la liste des commandes à livrer, soit des commandes d'enlèvement au magasin.

Cas d'utilisation : Accès à la page d'accueil

Développé par Stéphane Bausseron

Explication technique

```
//Requête pour afficher les commandes statut A préparer
$req_sqlPreLi = "SELECT * FROM Commandes
INNER JOIN Etats ON Commandes.id_etat = Etats.id_etat
INNER JOIN Couleurs ON Commandes.id_couleur = Couleurs.id_couleur
INNER JOIN Types ON Commandes.id_type = Types.id_type
INNER JOIN Clients ON Commandes.id_client = Clients.id_client
INNER JOIN Statuts ON Commandes.id_statut = Statuts.id_statut
WHERE Commandes.id_statut = 1
ORDER BY COALESCE(dateLivraison, dateEnlevement, dateContrat) ASC";
$req = $DB->prepare($req_sqlPreLi);
$req->execute();
$req_preList = $req->fetchAll();
```

```
//Requête pour compter le nombre de Commandes
$req_sqlComCount = "SELECT count(id_commande) FROM Commandes";
$req = $DB->prepare($req_sqlComCount);
$req->execute();
$req_commandes = $req->fetchAll();
$resultCommandes = $req_commandes[0][0];
```

Voici la requête « SQL » avec jointures permettant de récupérer toutes les commandes ayant le statut à préparer qui correspond à l'« id_statut = 1 » et avec toutes les jointures nécessaires pour afficher les informations liées à ces commandes avec également un ordre de tri directement inclus dans la requête ce qui permet d'avoir les commandes triées par date de la plus proche à la plus éloignée, je me sers également de l'objet « PDO » par « \$DB » pour la connexion à la base de données suivie du « prepare » pour une requête préparée on peut donc exécuter la requête et enfin le « fetchAll » pour récupérer toutes les données.

Voici ici la requête SQL permettant de compter le nombre de commandes enregistrées en base de données

Cas d'utilisation : Accès à la page d'accueil

Développé par Stéphane Bausseron

Explication technique

```
<?php
if (empty($req_preList)) {
    echo "<p class='row col-5 offset-1 '>il n'y a pas de commandes à préparer dans la liste ! </p>";
}else{
    foreach ($req_preList as $rpl){
        $id_commande = $rpl["id_commande"];
        $id_statut = $rpl["id_statut"];
        $nomClient = $rpl["nomClient"];
        $prenomClient = $rpl["prenomClient"];
        $dateLivraison = $rpl["dateLivraison"];
        $dateEnlevement = $rpl["dateEnlevement"];
        $dateContrat = $rpl["dateContrat"];
        $libBouquet = $rpl["libBouquet"];
        $libCouleur = $rpl["libCouleur"];
        $libEtat = $rpl["libEtat"];
        $prixVente = $rpl["prixVente"];
        $quantiteCommande = $rpl["quantiteCommande"];
        $prixCommande = $quantiteCommande * $prixVente ;

        ?>
        <tr>
            <td>
                <div class="text-center">
                    <h6 class="text-sm font-weight-bold mb-0 "><?= $nomClient ?></h6>
                </div>
            </td>
```

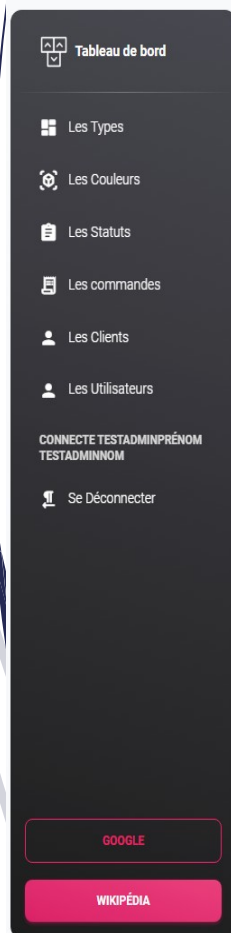
```
<?php
session_start();
include_once "_db/connexionDB.php";
$title = "Accueil";
$titleHeader = "Statistiques ALBIZIA";
if($_SESSION["user"]["activer"] == 1){
    include_once "v_header.php";
}elseif($_SESSION["user"]["activer"] == 2){
    include_once "v_deconnexion.php";
}elseif (!isset($_SESSION["user"]["id"])){
    header( header: "Location: index.php");
    exit();
}
```

J'ai également utilisé les session afin d'autoriser l'accès aux pages uniquement si l'utilisateur possède une session.

Le code à gauche montre la boucle « foreach » permettant d'afficher les données dynamiquement provenant de la base de données dans la liste des commandes à préparer

Cas d'utilisation : Accès à la page d'accueil

Développé par Stéphane Bausseron



Statistiques ALBIZIA



Types
3

Liste des Types



Couleurs
3

Liste des Couleurs



Commandes
5

Liste des Commandes



Clients
3

Liste des clients



COMMANDES A PREPARER (3) Le : 01/08/2023

Nom :	Prénom :	Date Heure :	Type :	Couleur :	Etat :	Prix Total :	
testNomClient1	testPrénomClient1	01/08/2023 18:40:00	bouquet multi-couleur	rouge et vert	non payée	38.45 €	
testNomClient3	testPrénomClient3	05/09/2023 09:30:00	bouquet multi-couleur	rose	contrat	76.9 €	
testNomClient3	testPrénomClient3	05/10/2023 09:30:00	bouquet multi-couleur	rose	contrat	76.9 €	

Voici le résultat de la page d'accueil après la connexion de l'administrateur. On peut voir que l'administrateur a un tableau de bord lui permettant d'accéder à toutes les interfaces de l'application, l'autre partie reste la même pour les utilisateurs, il n'a pas le tableau de bord mais juste un bouton pour se déconnecter. Les 4 cartes sont des liens pour accéder à l'interface souhaitée, la liste des commandes à préparer affiche toutes les commandes avec le statut à préparer et un icon urgent pour les commandes à la date du jour. Nous pouvons donc cliquer sur l'icon pour accéder au détail de la commande. Dans la partie suivante nous allons voir après le clic sur l'icône de la commande souhaitée le détail de celle-ci.

Cas d'utilisation : Détails de la commande

Développé par Stéphane Bausseron

DÉCONNEXION

Information Commande

Détail Client

Nom :	Prénom :	Adresse :	Ville :	Code Postal :	Email :	Téléphone :
testNomClient1	testPrénomClient1	test adresse client1	testVilleClient1	85000	testmailclient1@mail.fr	01.01.01.01.01

Détail Commande

Date Enlèvement :	Type :	Statut Bouquet :	Couleur :	Statut Couleur :	Statut :
01/08/2023 18:40:00	bouquet multi-couleur	été 2023	rouge et vert	tendance été	à préparer
Prix De Vente :	Quantité :	Prix Total :	Etat :		
38.45	1	38.45	non payée		

IMPRIMER

FIN PREPARATION

RETOUR

Nous voici sur la page du détail de la commande après le clique dans la liste sur l'icône de la page d'accueil, on peut voir le détail de toute la commande et en l'occurrence ici il s'agit d'un enlèvement au magasin stipulé par la date (date Enlèvement) avec un bouton « FIN PREPARATION » qui basculera la commande dans l'état « attente enlèvement », s'il s'agissait d'une date de livraison l'état de la commande après le clique « FIN PREPARATION » bascule la commande dans l'état « à livrer » . Il y a aussi un bouton « IMPRIMER » suite à la demande du client qui dans certain cas doit pouvoir imprimer le détail de la commande. Une fois la commande préparée l'utilisateur clique donc sur le bouton « FIN PREPARATION » la commande partira alors dans sa liste respective dans notre cas liste des commandes enlèvement.

Cas d'utilisation : Détails de la commande

Développé par Stéphane Bausseron

Explication technique

```
<div class="col-sm-2">
  <?php
  if (isset($dateLivraison) && $id_statut == 1) { ?>
    <a href="_controleurs/c_modifStatutLiv.php?id=<? $clientInfos["id_commande"] ?>">
      <button type="submit" class="btn bg-gradient-success w-100 my-4 mb-1"
        name="preparer">fin preparation</button></a>
    <?php
  } elseif (isset($dateEnlevement) && $id_statut == 1) {
    ?>
    <a href="_controleurs/c_modifStatutEnv.php?id=<? $clientInfos["id_commande"] ?>">
      <button type="submit" class="btn bg-gradient-success w-100 my-4 mb-1"
        name="preparer">fin preparation</button></a>
    <?php
  }
  ?>
</div>
```

Ci dessus le code HTML avec la partie PHP me permettant d'identifier la date ou de livraison ou d'enlèvement et au clique du bouton va transmettre le formulaire au bon contrôleur avec l'id de la commande qui lui pourra traiter la demande et mettre à jour le statut.

Ci dessous le contrôleur qui reçoit l'id de la commande est qui va avec une requête « SQL préparée » mettre à jour le statut de cette commande grâce à l'id envoyé par le formulaire à l'état correspondant c'est à dire pour notre cas on peut voir « id_statut = 4 » qui correspond à l'état « attente enlèvement », on exécute cette requête suivie d'un « fetch » car on ne traite qu'une commande et on redirige l'utilisateur sur la page d'accueil. L'utilisateur peut donc continuer une autre commande dans la liste « à préparer » de la page d'accueil, dans notre cas nous allons aller sur la liste Attente Enlèvements pour vérifier si notre commande apparaît.

```
<?php
session_start();
include_once "../db/connexionDB.php";

if($_SESSION["user"]["activer"] == 2) {
  include_once "../v_deconnexion.php";
} elseif (!isset($_SESSION["user"]["id"])) {
  header( header: "Location: ../index.php");
  exit();
}

if(isset($_GET["id"]) AND !empty($_GET["id"])) {
  $getid = $_GET['id'];
  $updateStatutEnv = $DB->prepare( query: 'UPDATE `Commandes` SET `id_statut` = 4 WHERE `id_commande`=?');
  $updateStatutEnv->execute(array($getid));
  $statutInfosEnv = $updateStatutEnv->fetch();
  header( header: "Location: ../v_accueil.php");
  exit();
}
```


15/32

Cas d'utilisation : Liste Attente Enlèvements

Développé par Stéphane Bausseron

Explication technique

DÉCONNEXION



Liste Attente Enlèvements

Nom :	Prénom :	Ville :	Date Enlèvement :	Type :	Couleur :	Statut :	Etat :	Prix Total :	Détail :	Finaliser :
testNomClient1	testPrénomClient1	testVilleClient1	01/08/2023 18:40:00	bouquet multi-couleur	rouge et vert	enlèvement	non payée	38.45 €		

IMPRIMER

RETOUR

Notre commande apparaît bien dans la liste des commandes Enlèvements ,on peut voir encore l'icône urgent car il s'agit toujours de la date du jour et un état non payée ,l'icône finaliser qui au clique de l'utilisateur va basculer le statut de la commande à finalisée et à l'état payée, à la demande du client j'ai également intégré le bouton imprimer pour certain cas , voyons au clique de l'icône finaliser.

16/32

Cas d'utilisation : Liste des Commandes

Développé par Stéphane Bausseron

DÉCONNEXION



Liste Attente Enlèvements

il n'y a pas de commandes prêtes en attente Enlèvement !

Nom :	Prénom :	Ville :	Date Enlèvement :	Type :	Couleur :	Statut :	Etat :	Prix Total :	Détail :	Finaliser :
-------	----------	---------	-------------------	--------	-----------	----------	--------	--------------	----------	-------------

IMPRIMER

RETOUR

DÉCONNEXION



Liste des Commandes

Nom :	Prénom :	Ville :	Date Heure :	Type :	Couleur :	Statut :	Etat :	Prix Total :			
testNomClient1	testPrénomClient1	testVilleClient1	01/08/2023 18:40:00	bouquet multi-couleur	rouge et vert	finalisée	payée	38.45 €	✓	✎	✖
testNomClient3	testPrénomClient3	testVilleClient3	05/08/2023 09:30:00	bouquet multi-couleur	rose	à livrer	contrat	76.9 €	🚚	✎	✖
testNomClient3	testPrénomClient3	testVilleClient3	05/09/2023 09:30:00	bouquet multi-couleur	rose	à préparer	contrat	76.9 €	📅	✎	✖
testNomClient3	testPrénomClient3	testVilleClient3	05/10/2023 09:30:00	bouquet multi-couleur	rose	à préparer	contrat	76.9 €	📅	✎	✖

IMPRIMER

RECHERCHER

AJOUTER

RETOUR

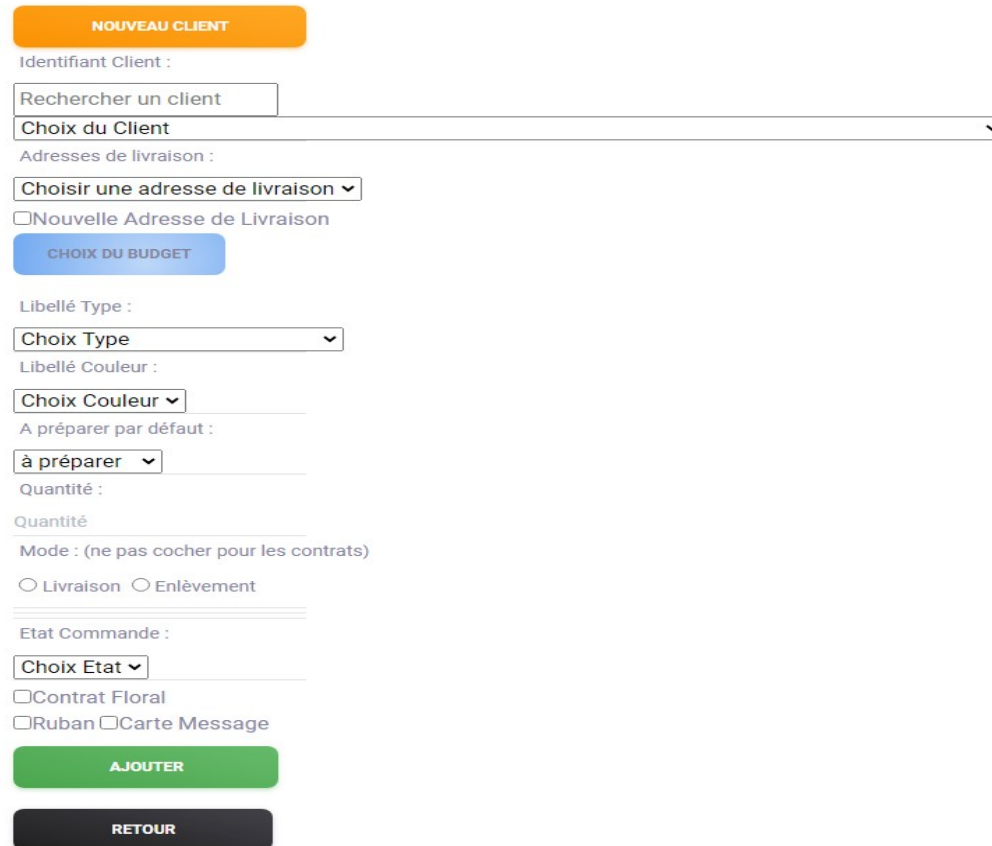
Du coup la liste ne contient plus de commandes en attente enlèvement dans notre cas on en avait qu'une il y aurait les autres à suivre mais la notre à disparut voyons maintenant comment elle apparaît dans la liste des commandes.

Voici la page liste des commandes qui affiche toutes les commandes et leurs états ainsi que la modification ou la suppression (uniquement connecté en tant qu'administrateur sinon l'utilisateur n'a pas les icônes correspondants), la commande est passé au statut finalisée avec un état payée

Cas d'utilisation : Ajout de commande

Développé par Stéphane Bausseron

Ajouter une commande



NOUVEAU CLIENT

Identifiant Client :

Rechercher un client

Choix du Client

Adresses de livraison :

Choisir une adresse de livraison

☐ Nouvelle Adresse de Livraison

CHOIX DU BUDGET

Libellé Type :

Choix Type

Libellé Couleur :

Choix Couleur

A préparer par défaut :

à préparer

Quantité :

Quantité

Mode : (ne pas cocher pour les contrats)

☐ Livraison ☐ Enlèvement

Etat Commande :

Choix Etat

☐ Contrat Floral

☐ Ruban ☐ Carte Message

AJOUTER

RETOUR

Nous voici sur la page « ajout commande », l'utilisateur peut ajouter un nouveau client en cliquant sur le bouton il accèdera directement au formulaire d'ajout client.

Si le client est déjà enregistré il peut le rechercher dans le champ de recherche en entrant les premières lettres de son nom et il pourra ainsi le trouver rapidement dans la liste déroulante.

Nous avons ensuite une autre liste déroulante pour le choix de l'adresse de livraison (le client peut avoir plusieurs adresses de livraisons qu seront affichées dans la liste), il peut aussi créer une adresse différente, dans ce cas en cochant la case les champs adresse, cp, ville apparaîtront et l'utilisateur ne pourra plus sélectionner une adresse dans la liste mais celle ci sera prise en compte.

A la suite le bouton « choix du budget » qui au clique fera apparaître une pop-up pour renseigner le montant se qui bloquera dans la liste « choix Type » tous les montants supérieur à celui saisi suivi de la liste « choix couleur » et la liste des statuts qui sera « à préparer » par défaut, le champ quantité qui s'il est égal à 0 ne valide pas la commande. Le mode : Livraison ou Enlèvement qui affichera un champ de sélection de date et heure.

La liste « choix état » qui possède 3 états « payée, non payée, contrat », si contrat alors la case à cocher fera apparaître un champ nombre de mois et un autre de sélection de date et heure, suivie des 2 autres options « ruban et carte message » pour ajout de texte.

Cas d'utilisation : Ajout de commande

Développé par Stéphane Bausseron

DÉCONNEXION

Ajouter un client

Nom	Nom du client :
Prénom	Prénom du client :
Email	Email du client :
Téléphone	Téléphone du client :
Adresse	Adresse du client :
Code Postal	Code Postal du client :
Ville	Ville du client :
0 = désactiver/ 1 = activer	Activer le client :

VALIDER

RETOUR

```
// le formulaire a été envoyé
// on vérifie que tous les champs requis sont remplis
if(!empty($_POST))
{
    extract( &array: $_POST);
    $valid = true;
    // Le formulaire est complet
    // On récupère les données en les protégeant
    if(isset($_POST["valider"])){
        $nomClient = strip_tags($_POST["nomClient"]);
        $prenomClient = strip_tags($_POST["prenomClient"]);
        $mailClient = strip_tags($_POST["mailClient"]);
        $telClient = strip_tags($_POST["telClient"]);
        $adresseClient = strip_tags($_POST["adresseClient"]);
        $cpClient = strip_tags($_POST["cpClient"]);
        $villeClient = strip_tags($_POST["villeClient"]);
        $activerClient = strip_tags($_POST["activerClient"]);
        if($valid == true){
```

Voici la page « Ajout Client » après le clique sur le bouton « nouveau client » l'utilisateur est redirigé sur ce formulaire.

On peut voir l'extrait du code me permettant de vérifier les informations saisies par l'utilisateur, la condition « if(!empty(\$_POST)) » permet de vérifier que les champs du formulaire sont remplis, la condition « if(isset) » permet de vérifier que le formulaire a bien été validé la fonction « strip_tags » fourni par « php » est utilisé ici afin de supprimer les caractères spéciaux se qui évite des injection de code dans les champs de saisies. Ensuite la requête « SQL » préparée (pour éviter les injections SQL) permettant de persister en base de données.

```
// on enregistre en bdd
$sql = "INSERT INTO Clients(nomClient, prenomClient,
    mailClient, telClient, adresseClient,
    cpClient, villeClient, activerClient)
    VALUES (:nomClient, :prenomClient,
        :mailClient, :telClient, :adresseClient,
        :cpClient, :villeClient, :activerClient)";

$query = $DB->prepare($sql);
$query->bindValue( param: ":nomClient", $nomClient, type: PDO::PARAM_STR);
$query->bindValue( param: ":prenomClient", $prenomClient, type: PDO::PARAM_STR);
$query->bindValue( param: ":mailClient", $_POST["mailClient"], type: PDO::PARAM_STR);
$query->bindValue( param: ":telClient", $telClient, type: PDO::PARAM_STR);
$query->bindValue( param: ":adresseClient", $adresseClient, type: PDO::PARAM_STR);
$query->bindValue( param: ":cpClient", $cpClient, type: PDO::PARAM_INT);
$query->bindValue( param: ":villeClient", $villeClient, type: PDO::PARAM_STR);
$query->bindValue( param: ":activerClient", $activerClient, type: PDO::PARAM_INT);
$query->execute();
```

Cas d'utilisation : Ajout de commande

Développé par Stéphane Bausseron

Identifiant Client :

te

testNomClient1 testPrénomClient1 - 01.01.01.01.01 / test adresse client1, 85000 testVilleClient1
 testNomClient1 testPrénomClient1 - 01.01.01.01.01 / test adresse client1, 85000 testVilleClient1
 testNomClient2 testPrénomClient2 - 02.02.02.02.02 / test adresse client2, 85400 testVilleClient2
 testNomClient3 testPrénomClient3 - 03.03.03.03.03 / test adresse client 3, 85700 testVilleClient3

```
function rechercherClient(searchTerm) {
  var idClientSelect = document.getElementById("idClient");
  var options = idClientSelect.options;
  var regex = new RegExp(searchTerm, "i"); // Expression régulière pour la recherche (ignorer la casse)

  for (var i = 0; i < options.length; i++) {
    var clientName = options[i].text;

    if (regex.test(clientName)) {
      options[i].style.display = "block"; // Afficher l'option si elle correspond à la recherche
    } else {
      options[i].style.display = "none"; // Masquer l'option si elle ne correspond pas à la recherche
    }
  }

  // Réinitialiser la liste déroulante en sélectionnant le premier client disponible après la recherche
  idClientSelect.selectedIndex = getFirstVisibleOptionIndex(options);
}
```

En premier le champ de la recherche en entrant 2 lettres du nom du client la liste déroulante affiche le résultat, il n'y a plus qu'à sélectionner le client souhaité.

Ensuite la fonction « JavaScript » qui me permet d'exécuter la recherche en fonction du champ saisi par l'utilisateur.

Puis une balise input qui va appeler la fonction à la saisie de l'utilisateur.

```
<input type="text" id="searchClient" oninput="rechercherClient(this.value)" placeholder="Rechercher un client">
<select name="id_client" id="idClient" onchange="afficherAutresAdressesLivraison(this)" required>
  <option value="">Choix du Client</option>
  <?php
    $req_sqlCl = "SELECT id_client, nomClient, prenomClient, telClient, adresseClient, cpClient, villeClient FROM Clients";
    $req = $DB->prepare($req_sqlCl);
    $req->execute();
    while ($req_clients = $req->fetch()){
      echo '<option value="'. $req_clients["id_client"] . '>' . $req_clients["nomClient"]
        . ' ' . $req_clients["prenomClient"] . ' - ' . $req_clients["telClient"] . ' / ' . $req_clients["adresseClient"]
        . ' ' . $req_clients["cpClient"] . ' ' . $req_clients["villeClient"] . '</option>';
    }
  ?>
</select>
```


Cas d'utilisation : Ajout de commande

Développé par Stéphane Bausseron

Identifiant Client :

Rechercher un client

testNomClient1 testPrénomClient1 - 01.01.01.01.01 / test adresse client1, 85000 testVilleClient1 ▼

Adresses de livraison :

Choisir une adresse de livraison ▼

Choisir une adresse de livraison

1 - nouvelle adresse client1 - 85370 nouvelleVilleClient1

```
function afficherAutresAdressesLivraison(selectClient) {
    var clientId = selectClient.value;
    var adresseLivraisonSelect = document.getElementById("adressesLivraison");
    var checkBox = document.getElementsByName("adresseLivraisonDifferent")[0];
    var labelCheckbox = document.getElementById("labelAdresseLivraisonDifferent");

    // Vérifier si un client est sélectionné
    if (clientId !== "") {
        // Envoyer une requête AJAX pour récupérer les adresses de livraison associées au client
        $.ajax({
            url: "get_adressesLivraison.php",
            method: "POST",
            data: { clientId: clientId },
            dataType: "json",
            success: function(response) {
                // Mettre à jour la liste déroulante des adresses de livraison avec les données reçues
                adresseLivraisonSelect.innerHTML = '<option value="">Choisir une adresse de livraison</option>';
                var uniqueAddresses = []; // Tableau pour stocker les adresses uniques
            }
        });
    }
}
```

L'utilisateur a sélectionné le client puis dans « adresse de livraison » à choisi une adresse de livraison déjà enregistrée différente de l'adresse du client se qui ne rend plus possible de cocher la case « nouvelle adresse de livraison ».

Ensuite un extrait de la fonction « JavaScript » permettant d'afficher les adresses de livraison liées au client sélectionné au préalable sans recharger la page grâce à une requête « AJAX » Puis la requête SQL et le renvoi de données en « JSON ».

```
// Effectuer la requête pour récupérer les adresses de livraison associées au client
$sql = "SELECT AdresseLivraisons.id_adresseLiv, AdresseLivraisons.adresseLiv,
        AdresseLivraisons.cpliv, AdresseLivraisons.villeLiv
        FROM Commandes
        INNER JOIN Clients ON Commandes.id_client = Clients.id_client
        INNER JOIN AdresseLivraisons ON Commandes.id_adresseLiv = AdresseLivraisons.id_adresseLiv
        WHERE Commandes.id_client = :clientId";

$query = $DB->prepare($sql);
$query->bindValue( param: ":clientId", $clientId, type: PDO::PARAM_INT);
$query->execute();
$adressesLivraison = $query->fetchAll( mode: PDO::FETCH_ASSOC);

// Retourner les adresses de livraison au format JSON
echo json_encode($adressesLivraison);
```


21/32

Cas d'utilisation : Ajout de commande

Développé par Stéphane Bausseron

Identifiant Client :

Rechercher un client

testNomClient2 testPrénomClient2 - 02.02.02.02.02 / test adresse client2, 85400 testVilleClient2 ▼

Adresses de livraison :

Choisir une adresse de livraison ▼

☒ Nouvelle Adresse de Livraison

Adresse :

nouvelle adresse client2

Code postal :

44000

Ville :

Nantes

```
<div class="row">
  <div class="text-start">
    <input type="checkbox" name="adresseLivraisonDifferent" onclick="afficherAdresseLivraison()">Nouvelle Adresse de Livraison
  </div>
</div>
<div id="adresseLivraisonFields" style="...">
  <div class="form-group mb-1">
    <div class="text-start">
      <label class="col-md-6 p-0">Adresse :</label>
      <div class="col-md-2 border-bottom p-0">
        <input name="adresseLivraison" type="text" placeholder="Adresse de livraison"
          class="form-control p-0 border-0 autocomplete="off">
      </div>
    </div>
  </div>
</div>
```

l'utilisateur n'a pas sélectionné d'adresse de livraison dans la liste il peut donc cocher la case pour faire apparaître les champs nécessaire afin d'ajouter une nouvelle adresse qui sera enregistrée en base de données dans la table « AdresseLivraisons » et reliée à la table « Commandes » par son id, on peut voir ici l'extrait de code « HTML » avec une fonction « JavaScript » permettant d'afficher les champs requis pour ajouter la nouvelle adresse.

Cas d'utilisation : Ajout de commande

Développé par Stéphane Bausseron

localhost indique

Veuillez saisir le montant du budget :

CHOIX DU BUDGET

Libellé Type :

Choix Type
Choix Type
bouquet rond 29.95 €
couronne fleurie 49.55 €
bouquet multi-couleur 38.45 €

Au clique sur le bouton de « choix du budget » une pop-up apparaît avec la demande de saisie du montant, après saisie du nombre (40 ici pour l'exemple) et validation de celle ci on peut voir que dans la liste « choix Type » la ligne « couronne fleurie 49.55€ » est grisée et ne peut donc pas être sélectionnée.

La fonction « saisirBudget() » permet d'afficher la pop-up et de récupérer la valeur.

Puis la fonction « filtrerTypes(budget) » avec la variable budget en paramètre va permettre de désactiver les montants supérieurs à celle ci.

```
function filtrerTypes(budget) {
    var selectType = document.getElementById("id_type");
    var options = selectType.options;

    for (var i = 0; i < options.length; i++) {
        var prixVente = parseFloat(options[i].getAttribute("data-prix"));

        if (isNaN(prixVente) || prixVente > budget) {
            options[i].disabled = true;
        } else {
            options[i].disabled = false;
        }
    }
}
```

```
function saisirBudget() {
    var budget = parseFloat(prompt("Veuillez saisir le montant du budget :"));

    if (isNaN(budget)) {
        // Gérer le cas où la valeur saisie n'est pas un nombre
        return;
    }

    filtrerTypes(budget);
}
```

Cas d'utilisation : Ajout de commande

Développé par Stéphane Bausseron

Libellé Couleur :

Choix Couleur ▼
 Choix Couleur
 rouge et vert
 jaune et violet
 rose

ut :

A préparer par défaut :

à préparer ▼
 à préparer
 à livrer
 enlèvement
 finalisée

ocher

Quantité

Ensuite une liste déroulante pour le « choix couleur », liste enregistrée au préalable à la réception de la marchandise comme pour les « Types » suivi de la liste des « statuts » qui sera « à préparer » par défaut si on ne sélectionne rien dans la liste ,et du champ « quantité » avec une fonction qui va permettre de valider si la quantité est différente de 0 , on attache cette fonction à la validation du formulaire permettant d'avertir l'utilisateur que la quantité n'a pas été saisie. Pour la liste des « statuts » la condition avant la requête pour mettre par défaut de « \$id_statut = 1 » qui correspond au statut « à préparer » et dans d'autre cas pouvant sélectionner un autre statut de la liste déroulante.

```
// Récupérer le champ quantiteCommande
const quantiteInput = document.getElementById("quantiteInput");

// Fonction pour vérifier la valeur avant de soumettre le formulaire
function validerQuantite() {
  const quantiteValue = parseInt(quantiteInput.value);

  // Vérifier si la valeur est différente de 0
  if (quantiteValue === 0) {
    alert("La quantité ne peut pas être égale à 0.");
    return false; // Empêcher la soumission du formulaire
  }

  // La quantité est valide, soumission du formulaire
  return true;
}

// Attacher la fonction validerQuantite à l'événement de soumission du formulaire
document.querySelector("form").addEventListener("submit", validerQuantite);
```

```
<label class="col-md-6 p-0">A préparer par défaut :</label>
<div class="col-md-2 border-bottom p-0">
  <select name="id_statut">
    <?php
      if (empty($id_statut)){
        $id_statut = 1;
      }
      $req_sqlSt = "SELECT id_statut, libStatut FROM Statuts";
      $req = $DB->prepare($req_sqlSt);
      $req->execute();
      while ($req_statuts = $req->fetch()){
        echo '<option value="'. $req_statuts["id_statut"]. '>'. $req_statuts["libStatut"]. '</option>';
      }
    ?>
  </select>
</div>
```

Cas d'utilisation : Ajout de commande

Développé par Stéphane Bausseron

Mode : (ne pas cocher pour les contrats)

☒ Livraison ☐ Enlèvement

jj/mm/aaaa --:--

contrat

☒ Contrat Floral

Nombre de mois

Sélectionnez les dates : jj/mm/aaaa --:--

☐ Livraison ☐ Enlèvement

Etat Commande :

contrat

Choix Etat

payée

non payée

contrat

Pour le choix du mode (Livraison ou Enlèvement au magasin) après le choix du bouton radio un champ de date et heure est proposé, puis la liste déroulante de l'état de la commande (payée, non payée, contrat) pour l'état contrat comme je l'ai stipulé pour l'utilisateur les boutons radio en amont sont inutiles car pour les contrats une case à cocher après sa validation affichera un champ de saisie du nombre de mois, puis le champ date et heure, également 2 boutons radio afin de choisir le mode (Livraison ou Enlèvement au magasin).

Pour les contrats après la validation de la case à cocher, j'ai encore utilisé une fonction JavaScript afin d'afficher les champs souhaités. Pour pouvoir enregistrer les commandes du contrat saisi il a fallu que je parcoure le nombre de mois saisi par l'utilisateur afin de créer une commande pour chaque itération avec un décalage d'un mois.

```
<div class="text-start">
  <input type="checkbox" name="multipleDates" value="1" id="multipleDates" onclick="afficherDates();" /> Contrat Floral
  <div class="row" id="nombreMoisContainer" style="...">
    <input type="number" name="nombreMois" id="nombreMois" placeholder="Nombre de mois" />
  </div>
  <div id="datesContainer" style="...">
    <label>Sélectionnez les dates :</label>
    <input type="datetime-local" name="dates[]" multiple />
    <input type="radio" name="livraisonEnlevement" value="livraison" /> Livraison
    <input type="radio" name="livraisonEnlevement" value="enlevement" /> Enlèvement
  </div>
</div>
```

```
// On enregistre en base de données pour chaque date sélectionnée
foreach ($dates as $date) {
  $currentDate = new DateTime($date);
  $dateCommande = date( format: "Y-m-d" ); // Mettre à jour la date de commande pour chaque itération
  for ($i = 0; $i < $nombreMois; $i++) {
    $dateContrat = date( format: "Y-m-d H:i:s", strtotime( datetime: "+{$i} month", strtotime($date) ));
  }
}
```


Cas d'utilisation : Ajout de commande

Développé par Stéphane Bausseron

Voici la partie du code pour persister les commandes contrats récurrentes en base de données un des soucis été de convertir les dates au bon format avant l'insertion en base de données et inversement de les remettre dans le bon format pour les affichées à l'utilisateur. La première partie est une condition si les champs du formulaire « multipleDates » à été renseigné suivie du traitement identique à une commande lambda puis la boucle avec une incrémentation de la longueur du nombre de mois et à la fin on peut voir une méthode ajoutant 1 mois d'intervall.

```
}elseif ($valid == true && isset($_POST["multipleDates"])) {
    if (isset($_POST["multipleDates"])) {
        $nombreMois = $_POST["nombreMois"];
        $dates = $_POST["dates"];
        $dateCommande = date("Y-m-d");
        $livraisonEnlevement = $_POST["livraisonEnlevement"]; // Récupérer le choix livraison ou enlèvement
        $quantiteCommande = $_POST["quantiteCommande"];
        $id_type = $_POST["id_type"];
        $id_etat = $_POST["id_etat"];
        $id_couleur = $_POST["id_couleur"];
        $id_type = $_POST["id_type"];
        $id_client = $_POST["id_client"];
        $id_statut = $_POST["id_statut"];
        $ruban = isset($_POST["ruban"]) ? 1 : 0;
        $rubanText = isset($_POST["rubanText"]) ? $_POST["rubanText"] : "";
        $carteMessage = isset($_POST["carteMessage"]) ? 1 : 0;
        $messageText = isset($_POST["messageText"]) ? $_POST["messageText"] : "";
```

```
// On enregistre en base de données pour chaque date sélectionnée
foreach ($dates as $date) {
    $currentDate = new DateTime($date);
    $dateCommande = date("Y-m-d"); // Mettre à jour la date de commande pour chaque itération
    for ($i = 0; $i < $nombreMois; $i++) {
        // En fonction du choix de l'utilisateur (livraison ou enlèvement), on assigne la date à la bonne colonne
        if ($livraisonEnlevement === "livraison") {
            $dateLivraison = $currentDate->format("Y-m-d H:i:s");
            $dateEnlevement = null;
        } elseif ($livraisonEnlevement === "enlevement") {
            $dateEnlevement = $currentDate->format("Y-m-d H:i:s");
            $dateLivraison = null;
        }
        $dateContrat = date("Y-m-d H:i:s", strtotime("date: "+{$i} month", strtotime($date)));
        $sql = "INSERT INTO Commandes (dateCommande, dateLivraison, dateEnlevement, dateContrat, quantiteCommande,
        prixCommande, id_etat, id_couleur, id_type, id_client, id_statut, ruban, carteMessage)
        VALUES (:dateCommande, :dateLivraison, :dateEnlevement, :dateContrat, :quantiteCommande,
        :quantiteCommande * (SELECT prixVente FROM Types WHERE id_type = :id_type),
        :id_etat, :id_couleur, :id_type, :id_client, :id_statut, :rubanText, :messageText)";
        $query = $DB->prepare($sql);
        $query->bindValue("dateCommande", $dateCommande, PDO::PARAM_STR);
        $query->bindValue("dateLivraison", $dateLivraison, PDO::PARAM_STR);
        $query->bindValue("dateEnlevement", $dateEnlevement, PDO::PARAM_STR);
        $query->bindValue("dateContrat", $dateContrat, PDO::PARAM_STR); // Utilisation de la date sél
        $query->bindValue("quantiteCommande", $quantiteCommande, PDO::PARAM_STR);
        $query->bindValue("id_etat", $id_etat, PDO::PARAM_STR);
        $query->bindValue("id_couleur", $id_couleur, PDO::PARAM_STR);
        $query->bindValue("id_type", $id_type, PDO::PARAM_STR);
        $query->bindValue("id_client", $id_client, PDO::PARAM_STR);
        $query->bindValue("id_statut", $id_statut, PDO::PARAM_STR);
        $query->bindValue("rubanText", $rubanText, PDO::PARAM_STR);
        $query->bindValue("messageText", $messageText, PDO::PARAM_STR);
        $query->execute();

        // Passer à la prochaine date en ajoutant un mois
        $currentDate->add(new DateInterval("P1M"));
    }
}
```


Cas d'utilisation : Recherche commande

Développé par Stéphane Bausseron

Après l'ajout de commandes par l'utilisateur nous sommes redirigé sur la page d'accueil ou nous pouvons voir la liste des commandes « à préparer », avec le nombre et la date du jour. Dans notre exemple on voit les commandes classées par date et heure de la plus proche à la plus éloigné, pour les commandes en date du jour elles sont repérées par un icône « urgent » ce qui permet à l'utilisateur une organisation de ses tâches, il suffit de cliquer sur l'icône pour accéder aux détails de la commande et commencer sa préparation.

L'utilisateur a également accès à la liste de toutes les commandes avec un choix de tri (par nom du client, par dates, par prix, par statuts et par mode), il peut alors voir l'historique des commandes.

Dans notre exemple nous cliquons sur l'icône de la commande en date du jour.


 COMMANDES A PREPARER (5) Le : 12/08/2023						
Nom :	Prénom :	Date Heure :	Type :	Couleur :	Etat :	Prix Total :
testNomClient1	testPrénomClient1	12/08/2023 10:40:00	bouquet multi-couleur	jaune et violet	non payée	38.45 €
testNomClient2	testPrénomClient2	14/08/2023 11:45:00	bouquet multi-couleur	jaune et violet	non payée	38.45 €
testNomClient3	testPrénomClient3	19/08/2023 09:30:00	couronne fleurie	rouge et vert	non payée	49.55 €
testNomClient3	testPrénomClient3	05/09/2023 09:30:00	bouquet multi-couleur	rose	contrat	76.9 €
testNomClient3	testPrénomClient3	05/10/2023 09:30:00	bouquet multi-couleur	rose	contrat	76.9 €

Cas d'utilisation : Détail commande

Développé par Stéphane Bausseron


DÉCONNEXION

Information Commande



Détail Client

Nom :	Prénom :	Adresse :	Ville :	Code Postal :	Email :	Téléphone :
testNomClient1	testPrénomClient1	test adresse client1	testVilleClient1	85000	testmailclient1@mail.fr	01.01.01.01.01



Détail Commande

Date Enlèvement :	Type :	Statut Bouquet :	Couleur :	Statut Couleur :	Statut :
12/08/2023 10:40:00	bouquet multi-couleur	été 2023	jaune et violet	couleur vive	à préparer

Prix De Vente :	Quantité :	Prix Total :	Etat :
38.45	1	38.45	non payée

IMPRIMER

FIN PREPARATION

RETOUR

Voici après le clique sur l'icône de la page d'accueil l'affichage du détail de la commande grâce a son « id », on y retrouve les informations du client correspondants à la table « Clients » ainsi que les informations de la commande correspondants aux tables « Commandes, Statuts, Couleurs, Types, Etats, AdresseLivraisons ».

Dans notre cas il s'agit comme on peut le voir d'une « Date Enlèvement » le champ adresse de livraison n'apparaît donc pas , l'utilisateur peut ainsi préparer la commande identifiée au mode « Enlèvement ».

La commande préparée , il clique sur le bouton « FIN PREPARATION » ce qui le redirige sur la page d'accueil contenant la liste des commandes « à préparer ».

Cas d'utilisation : Liste commande

Développé par Stéphane Bausseron

De retour sur la liste des commandes « à préparer » comme nous pouvons le voir la liste contenait 5 commandes avec une commande en date du jour, cette commande a disparu de la liste « à préparer » pour s'afficher dans la liste des commandes « en attente Enlèvement », l'utilisateur peut ainsi gérer ses tâches en fonction des priorités, pour l'exemple nous avons qu'une seule commande « urgente » mais si il y en avait plusieurs l'utilisateur aurait pu continuer à la suite la préparation des commandes affichées par date et heure de priorité.

Nous allons aller voir la liste des commandes « Attente Enlèvement » afin de retrouver la commande que l'on vient de traiter.



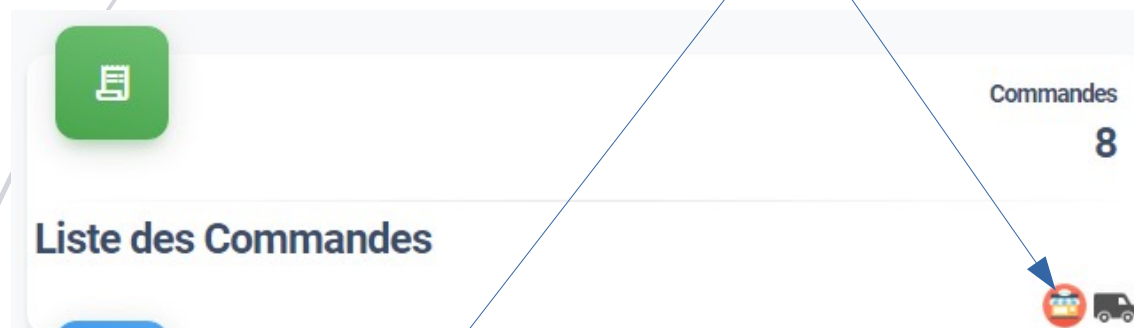
COMMANDES A PREPARER (4) Le : 12/08/2023

Nom :	Prénom :	Date Heure :	Type :	Couleur :	Etat :	Prix Total :	
testNomClient2	testPrénomClient2	14/08/2023 11:45:00	bouquet multi-couleur	jaune et violet	non payée	38.45 €	
testNomClient3	testPrénomClient3	19/08/2023 09:30:00	couronne fleurie	rouge et vert	non payée	49.55 €	
testNomClient3	testPrénomClient3	05/09/2023 09:30:00	bouquet multi-couleur	rose	contrat	76.9 €	
testNomClient3	testPrénomClient3	05/10/2023 09:30:00	bouquet multi-couleur	rose	contrat	76.9 €	

Commande attente Enlèvement

Développé par Stéphane Bausseron

Nous avons donc cliqué sur cet icône correspondant au mode « Attente Enlèvement » se trouvant sur la page d'accueil dans la carte « Liste des commandes » ce qui nous redirige sur la page « Liste Attente Enlèvements »



Liste Attente Enlèvements

Nom :	Prénom :	Ville :	Date Enlèvement :	Type :	Couleur :	Statut :	Etat :	Prix Total :	Détail :	Finaliser :
testNomClient1	testPrénomClient1	testVilleClient1	12/08/2023 10:40:00	bouquet multi-couleur	jaune et violet	enlèvement	non payée	38.45 €		

[IMPRIMER](#) [RETOUR](#)

Nous voyons bien notre commande préalablement préparée qui est donc passé du statut « à préparer » au statut « enlèvement ». Il reste encore une étape pour finaliser cette commande (le paiement et l'enlèvement de celle ci par le client) .

On peut également constater que l'icône détail est toujours affichée en « urgent » ce qui n'aurait pas été le cas si la commande avait eu une date supérieur à la date du jour. Il ne reste plus qu'à cliquer sur le bouton « finaliser ».

Commande Finalisée

Développé par Stéphane Bausseron

Nous voici sur la page de la liste des commandes après avoir cliqué sur l'icône « finaliser » précédemment, on retrouve ici toutes les commandes avec les différents statuts, pour notre exemple notre commande à maintenant le statut « finalisée » avec l'état « payée » ainsi que l'icône « urgent » devenu l'icône « finalisée ».

On peut quand même toujours accéder au détail de la commande en cliquant sur ce même icône qui nous envoi sur la page détail mais le bouton « FIN PREPARATION » est désactivé.

On peut également voir 2 icônes supplémentaires (uniquement pour l'administrateur) qui permettent de modifier ou supprimer une commande, j'ai donc utilisé le « CRUD » dans diverses parties de l'application ci après les extrait de code .

Liste des Commandes

Nom :	Prénom :	Ville :	Date Heure :	Type :	Couleur :	Statut :	Etat :	Prix Total :			
testNomClient1	testPrénomClient1	testVilleClient1	12/08/2023 10:40:00	bouquet multi-couleur	jaune et violet	finalisée	payée	38.45 €			
testNomClient2	testPrénomClient2	testVilleClient2	14/08/2023 11:45:00	bouquet multi-couleur	jaune et violet	à préparer	non payée	38.45 €			
testNomClient3	testPrénomClient3	testVilleClient3	19/08/2023 09:30:00	couronne fleurie	rouge et vert	à préparer	non payée	49.55 €			
testNomClient3	testPrénomClient3	testVilleClient3	05/09/2023 09:30:00	bouquet multi-couleur	rose	à préparer	contrat	76.9 €			
testNomClient3	testPrénomClient3	testVilleClient3	05/10/2023 09:30:00	bouquet multi-couleur	rose	à préparer	contrat	76.9 €			

[IMPRIMER](#)[RECHERCHER](#)[AJOUTER](#)[RETOUR](#)

Extrait du CRUD

Développé par Stéphane Bausseron

Ci dessous la requête SQL UPDATE qui va mettre à jour au clique du bouton « finaliser » le statut à « finalisée » et l'état à « payée » on peut voir également que si l'utilisateur essaye d'entré sur la page sans être connecté il est redirigé sur la page de connexion « index.php ».

```
session_start();
include_once "../_db/connexionDB.php";

if($_SESSION["user"]["activer"] == 2) {
    include_once "../v_deconnexion.php";
}elseif (!isset($_SESSION["user"]["id"])){
    header( header: "Location: ../index.php");
    exit();
}

if(isset($_GET["id"]) AND !empty($_GET["id"])) {
    $getid = $_GET['id'];
    $updateEnvFinal = $DB->prepare( query: 'UPDATE `Commandes`
    SET `id_statut` = 9, `id_etat` = 1 WHERE `id_commande`=?');
    $updateEnvFinal->execute(array($getid));
    $statutEnvFinal = $updateEnvFinal->fetch();
    header( header: "Location: ../v_listEnlevement.php");
    exit();
}
```

Ci dessous la requête SQL DELETE qui va supprimer la commande grâce a son « id » et dans ce cas on peut voir que l'accès est réservé à l'administrateur « \$_SESSION['user']['activer'] == 1 », l'utilisateur est redirigé sur la page d'accueil sinon s'il n'est pas connecté sur la page connexion.

```
session_start();
include_once "../_db/connexionDB.php";

if ($_SESSION["user"]["activer"] == 1) {
    include_once "../v_deconnexion.php";
}elseif ($_SESSION["user"]["activer"] == 2) {
    header( header: "Location: ../v_accueil.php");
}elseif (!isset($_SESSION["user"]["id"])) {
    header( header: "Location: ../index.php");
    exit();
}

$id_commande = $_GET['id'];
$sql = "DELETE FROM `Commandes` WHERE id_commande = $id_commande";
$requete = $DB->query($sql);

if(isset($id_commande)){
    header( header: "Location: ../v_rechercher.php");
}else{
    header( header: "Location: ../v_rechercher.php");
}
```

Conclusion

- ▮ Je tiens à remercier les formateurs de l'ENI pour m'avoir transmis une partie de leur savoir faire du métier passionnant qu'est Développer Web ainsi que mon tuteur de stage pour sa bienveillance et de m'avoir fait confiance sur le projet Albizia qui a été pour moi une expérience formidable et très enrichissante car travaillant seul sur le projet, il a fallu beaucoup de recherche et d'investissement pour mener à bien ce projet.
- ▮ Ce projet m'a permis de mettre en pratique énormément de cours appris en formation et dans acquérir d'avantage avec un projet concret.
- ▮ Points principaux du projet :
- ▮ Maquettage de l'application
- ▮ Création de la base de données MySQL
- ▮ Mise en forme avec un framework Bootstrap
- ▮ Développement Front End (HTML, CSS, JAVASCRIPT)
- ▮ Développement Back End (PHP, SQL)
- ▮ Consignes de sécurité Front End (méthodes et conditions pour contrôle des champs utilisateurs).
- ▮ Consignes de sécurité Back End (conditions et requêtes préparées afin d'éviter les injections SQL).
- ▮ Tests Unitaire de l'application et présentation du projet finalisé.
- ▮ Déploiement sur un server FTP