

Document d'exploitation

- Le manuel d'utilisation

1. Inscription

Fonctionnalités :

- Permet aux utilisateurs de créer un compte en fournissant les informations suivantes :
 - **Pseudo**
 - **E-mail**
 - **Mot de passe**
 - **Vérification du mot de passe**
- Les données saisies sont validées avant d'être insérées dans la base de données.

Interface utilisateur

Resto Web Accueil Panier Connexion Inscription Non connecté

Inscription

Pseudo

Mail

Mot de passe

Verification du mot de passe

S'inscrire

Code :

Traitement PHP (dans `inscription.php`)

```
if ($submit){  
    //Rajouter des isset avec les erreurs liées  
    $loginUtil = $_POST['loginUtil'];  
    $mail = $_POST['mail'];  
    $pwd = $_POST['pwd'];  
    $pwd_check = $_POST['pwd_check'];  
  
    //rejouer execute si tous les isset sont ok  
    $error_message=add_user_db();  
}
```

On utilise des champs html, pour pouvoir récupérer les informations de l'utilisateur et les attribuer à des variables. Ces informations sont ensuite passées en session pour pouvoir les réutiliser sans passer par des requêtes SQL.

2. Connexion

Fonctionnalités :

- Permet aux utilisateurs de se connecter à leur compte avec leur **login** et **mot de passe** .
- Redirige l'utilisateur vers la page d'accueil en cas de succès.

Interface utilisateur

The screenshot displays a web application interface for 'Resto Web'. At the top, a dark navigation bar contains the site name 'Resto Web' and four menu items: 'Accueil', 'Panier', 'Connexion', and 'Inscription'. The 'Connexion' item is highlighted with a gold background. In the top right corner of the page, the text 'Non connecté' indicates the user's status. The main content area has a light purple background and features a central white login box with rounded corners and a subtle shadow. This box is titled 'Connexion' and contains two input fields labeled 'Login' and 'Password', followed by a 'Connexion' button.

Code :

Algorithme PHP (dans **connection.php**)

```

session_start();

// Initialisation de variables pour les messages d'erreur
$error_message = null;
$submit = isset($_POST['submit']);

if ($submit) {
    $login = isset($_POST['login']) ? $_POST['login'] : "";
    $password = isset($_POST['password']) ? $_POST['password'] : "";

    // Vérification que les champs ne sont pas vides
    if (!empty($login) && !empty($password)) {
        // Appeler la fonction pour vérifier l'utilisateur dans la base de données
        if (login_user_db($login, $password)) {
            // Si l'utilisateur est trouvé et le mot de passe est correct
            header("Location: ../../index.php"); // Redirection vers la page d'accueil
            exit();
        } else {
            // Message d'erreur si le login ou mot de passe est incorrect
            $error_message = "Login ou mot de passe incorrect.";
        }
    } else {
        // Message d'erreur si les champs sont vides
        $error_message = "Veuillez remplir tous les champs.";
    }
}

?>

```

Ce code en php indique que si le bouton connexion est cliqué, on récupère les login et password qui ont été mis en session.

On vérifie que les champs ne soient pas vide, si c'est le cas on vérifie à l'aide de la fonction login_user_db() que les login et password sont corrects et si c'est le cas on redirige vers la page d'accueil.

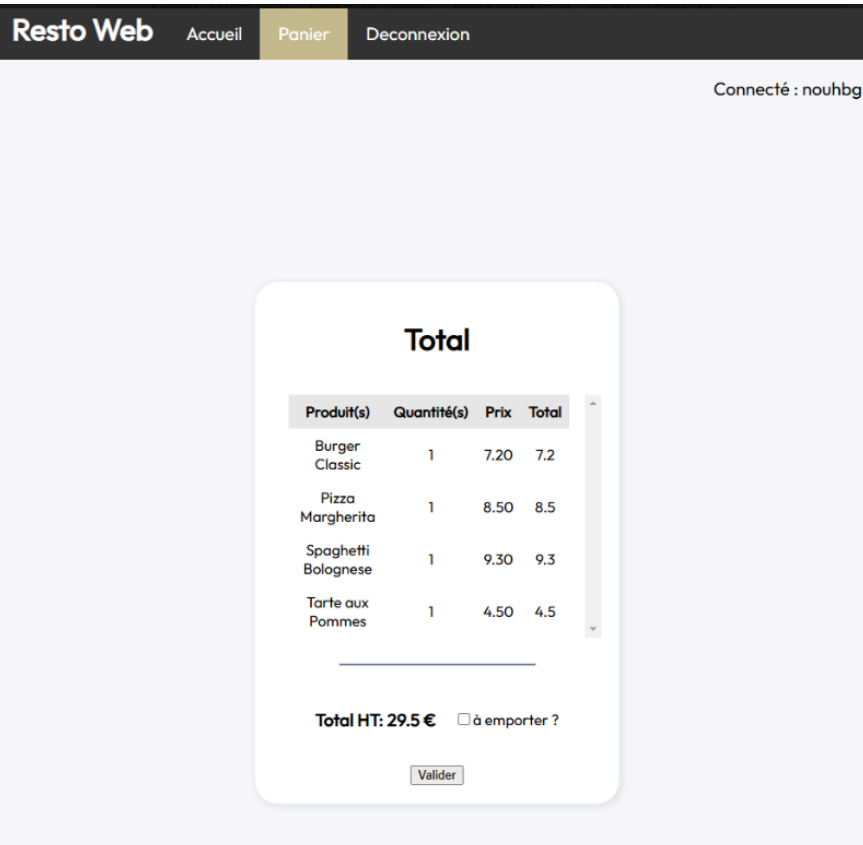
Dans le cas contraire, un message d'erreur s'affiche en disant que le login ou le mot de passe est incorrect.

3. Commande

Fonctionnalités :

- Affichez les produits dans le panier avec leurs informations (quantité, prix unitaire, total).
- Calculez le total HT pour le panier.
- Validez le panier en enregistrant la commande et ses lignes dans la base de données.

Interface utilisateur



Code :

Affichage des produits dans le panier (dans **panier.php**)

```

echo "<table>
    <tr>
        <th>Produit(s)</th>
        <th>Quantité(s)</th>
        <th>Prix</th>
        <th>Total</th>
    </tr>";
$j=0;
foreach($cart as $products){

    foreach($products as $product){

        $id=$product['productId'];
        $qt=$product['quantity'];
        $product=get_product($id);
        $lib=$product['libProduit'];
        $prix=$product['prixProHT'];
        $total=$prix*$qt;
        $totalHT+=$total;
        echo "<tr>
            <td><input type='hidden' name='idPro' . $j . '' value='' . $id . ''> . $lib . "</td>
            <td><input type='hidden' name='qtLigne' . $j . '' value='' . $qt . ''> . $qt . "</td>
            <td><input type='hidden' name='prixProHT' . $j . '' value='' . $prix . ''> . $prix . "</td>
            <td><input type='hidden' name='totalLigneHT' . $j . '' value='' . $total . ''> . $total . "</td>
        </tr>";

        $j++;
    }
}

echo "<input type='hidden' name='totalHT' value='' . $totalHT . ''>";
echo "</table>";

```

La page panier utilise la session pour récupérer les différents produits et leurs quantités pour pouvoir les afficher à l'aide d'une boucle foreach. Mais pour l'affichage sous forme de quadrillage, il est nécessaire d'utiliser un autre foreach.

Insertion de la commande (dans `insert_commande.php`)

```

1 <?php
2 function insert_commande($totalComTTC,$typeCom,$idUtil){
3     $dbh=connect_db();
4     $sql="INSERT INTO commande (etatCom, totalComTTC, typeCom, dateCom, heureCom, idUtil) VALUES (:etatCom, :totalComTTC, :typeCom, :dateCom, :heureCom, :idUtil)";
5     try{
6         $sth=$dbh->prepare($sql);
7         $sth->execute(array(':etatCom'=>0,':totalComTTC'=>$totalComTTC,':typeCom'=>$typeCom,':dateCom'=>date('Y-m-d'),':heureCom'=>date('H:i:s'),':idUtil'=>$idUtil));
8         $insertState=true;//Insertion réussie
9     }catch(PDOException $e){
10         die("Erreur dans la fonction insertCommande : ".$e->getMessage());
11         $insertState=false;//Insertion échouée
12     }
13     //On enregistre que l'on vient de créer dans la session pour s'en servir dans la page payer.php
14     $_SESSION['idCom']=$dbh->lastInsertId();
15
16     return $insertState; //Retour du statut de l'insertion
17 }
18
19
20
21 >

```

Lorsque le bouton commander est cliqué, une requête SQL s'exécute pour pouvoir ajouter dans la table commande les différents champs nécessaires.

La requête est préparée puis exécutée pour pouvoir éviter les injections sql. Si la requête génère une erreur, on utilise une variable qui est renseignée par PDOException pour renvoyer le message d'erreur de MySQL.

4. Paiement

Fonctionnalités :

- Permet à l'utilisateur d'entrer ses informations bancaires pour payer sa commande.
- Met à jour l'état de la commande dans la base de données après validation.

Interface utilisateur

Resto Web Accueil Panier Deconnexion

Connecté : nouhbg

Paiement de votre commande

Total de la commande : 32.45 €

N° de CB

Date expiration

CCV

Payer

Code :

Formulaire HTML (dans `payer.php`)

```
<?php  
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">  
  <input type="text" name="cb_number" placeholder="N° de CB" required="required">  
  <input type="text" name="exp_date" placeholder="Date expiration" required="required">  
  <input type="text" name="ccv" placeholder="CCV" required="required">  
  <input type="submit" name="submit" value="Payer">  
<?php  
if($paymentState==true){  
  echo "<p class='form_success_message'>Paiement effectué avec succès</p>";  
  echo "<p class='form_success_message'>Vous allez être redirigé vers la page de confirmation de commande</p>";  
}  
?>
```

Pour le paiement, le client renseigne les champs de la carte bleue en vérifiant que ces champs sont bien renseignés.

Mise à jour de la commande (dans `payer.php`)

```
try {
    $etatCom = "Calculée";
    $sql = "UPDATE commande SET etatCom = :etatCom WHERE idCom = :idCom;";
    $stmt = $pdo->prepare($sql);
    $stmt->execute([
        ':etatCom' => $etatCom,
        ':idCom' => $idCom
    ]);
}
```

Une fois le bouton payer cliquer, on fait une requête UPDATE pour la base de données pour pouvoir mettre à jour l'état de la commande.

Partie RestoSwing :

L'application **RestoSwing** est un programme développé en Java, spécialement conçu pour les restaurateurs. Elle leur permet de voir, gérer et suivre les commandes passées par les clients sur le site web. Il y a deux fenêtres principales : une qui montre la liste des commandes, et une autre qui affiche les détails d'une commande sélectionnée.

Quand l'application est lancée, la première fenêtre s'ouvre automatiquement et montre toutes les commandes en attente. Chaque ligne du tableau correspond à une commande, avec son numéro, la date, l'heure, le nombre de plats, le montant total et son état actuel. Cela permet au restaurateur de voir rapidement les commandes à traiter. Il peut ensuite cliquer sur le bouton **"Détails"** pour voir plus d'informations sur une commande.

Dans la fenêtre de détails, on peut voir le contenu de la commande choisie. Elle affiche le numéro de la commande, la date et l'heure de création, ainsi que le nom du client. Un tableau présente tous les plats commandés, avec leur nom, la quantité demandée et leur identifiant. Sur la droite, il y a trois boutons : **Accepter**, **Refuser** et **Prête**. En cliquant dessus, l'état de la commande est mis à jour automatiquement grâce à l'appel de l'api pour modifier le statut de la commande. Un bouton **"Revenir"** permet de revenir à la liste des commandes.

Extrait de code et des captures :

Page d'accueil :

RestoSwing

RestoSwing

Liste des commandes

ID	Date	Heure	Etat	Nb plats	Montant
192	30/11/2023	00H00	en attente	6	0.0
193	30/11/2023	00H00	en attente	0	26.0
194	30/11/2023	00H00	en attente	0	26.0
195	30/11/2023	00H00	en attente	0	27.0
196	06/12/2023	18H57	en attente	0	57.0
197	15/01/2024	14H21	en attente	1	13.0
198	15/01/2024	15H14	en attente	6	88.0
199	15/01/2024	15H35	en attente	3	85.8
200	15/01/2024	15H37	en attente	1	13.75
201	16/02/2024	16H52	en attente	2	38.5
202	16/02/2024	16H52	en attente	1	11.0
203	16/02/2024	17H43	en attente	3	34.65

Détails

Quitter

Détails commande :

RestoSwing

Détails d'une commande

ID commande

203

Date

16/02/2024 17H43

Login

jef

ID	Plat	Quantité
333	pizza Margherita	1
334	hamburger vegan	1
335	hot dog	2

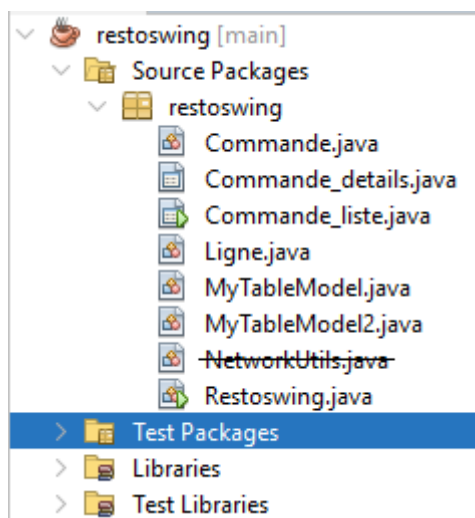
Accepter

Refuser

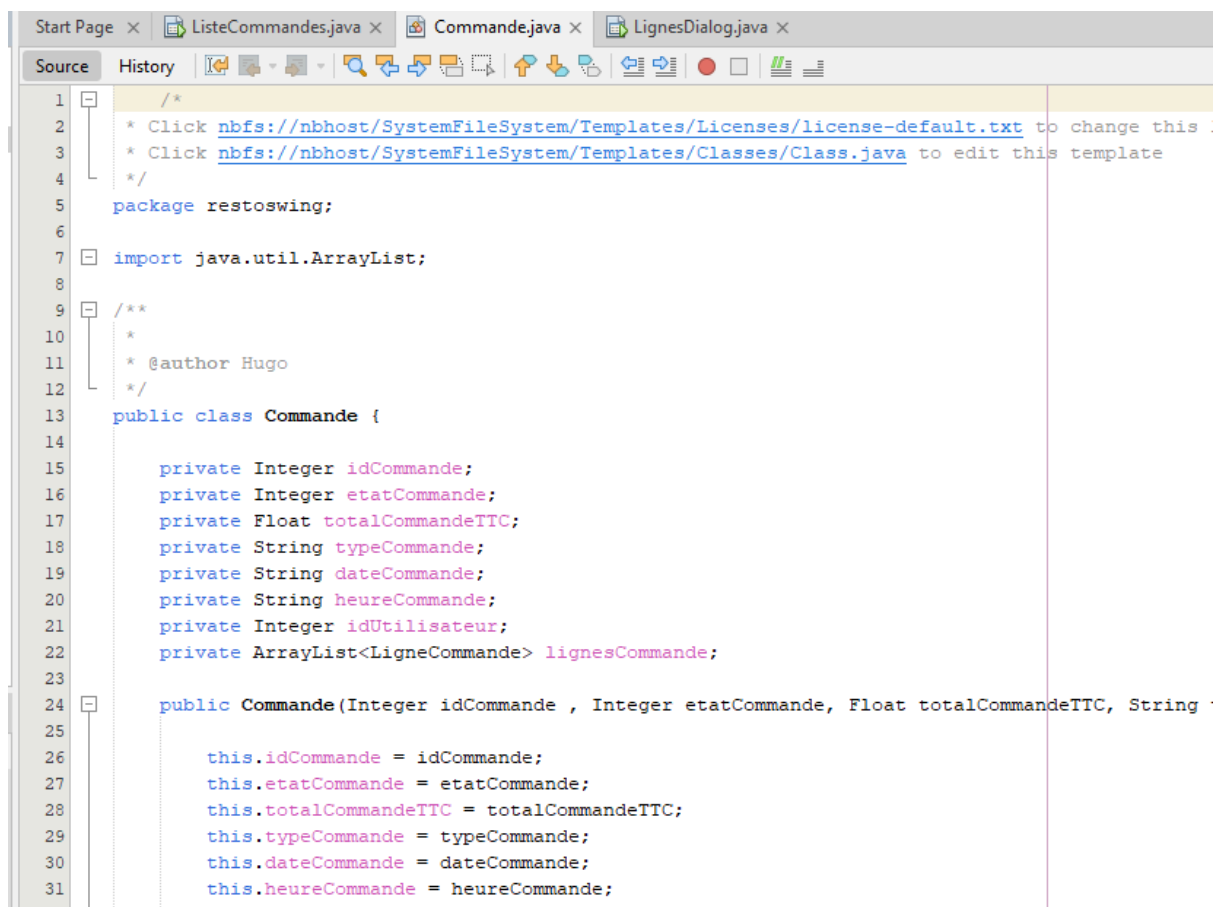
Prête

Revenir

Extrait de fichier :



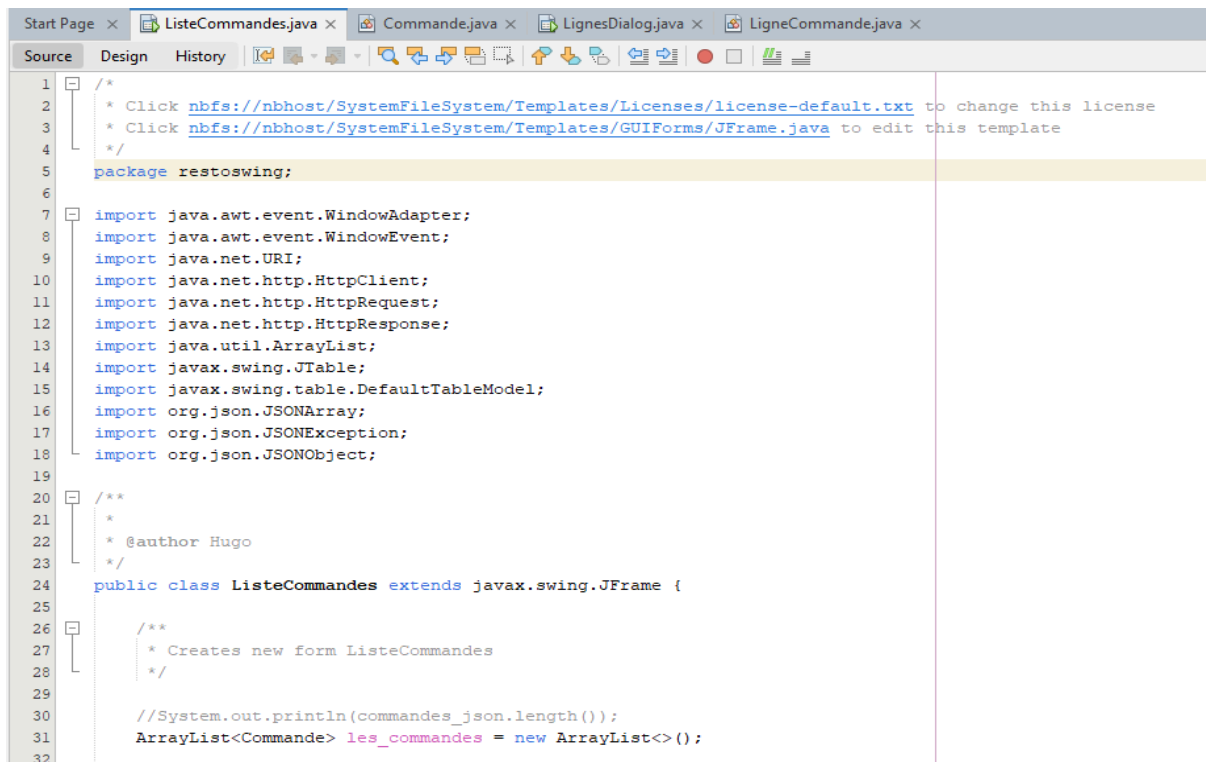
Commande.java :



LigneCommande.java :

```
Start Page x ListeCommandes.java x Commande.java x LignesDialog.java x LigneCommande.java x
Source History
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5   package restoswing;
6
7   /**
8    *
9    * @author Hugo
10   */
11  public class LigneCommande {
12
13      private Integer idCommande;
14      private Integer idProduit;
15      private Integer qteLigne;
16      private Float totalLigneHT;
17
18      public LigneCommande( Integer idCommande, Integer idProduit, Integer qteLigne, Float totalLigneHT) {
19
20          this.idCommande = idCommande;
21          this.idProduit = idProduit;
22          this.qteLigne = qteLigne;
23          this.totalLigneHT = totalLigneHT;
24      }
25
26      public Integer getIdCommande() {
27          return idCommande;
28      }
29
30      public void setIdCommande(Integer idCommande) {
31          this.idCommande = idCommande;
32      }
```

Extrait de la pages liste commande qui représente la page d'accueil:



```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template
4  */
5  package restoswing;
6
7  import java.awt.event.WindowAdapter;
8  import java.awt.event.WindowEvent;
9  import java.net.URI;
10 import java.net.http.HttpClient;
11 import java.net.http.HttpRequest;
12 import java.net.http.HttpResponse;
13 import java.util.ArrayList;
14 import javax.swing.JTable;
15 import javax.swing.table.DefaultTableModel;
16 import org.json.JSONArray;
17 import org.json.JSONException;
18 import org.json.JSONObject;
19
20 /**
21 *
22 * @author Hugo
23 */
24 public class ListeCommandes extends javax.swing.JFrame {
25
26     /**
27      * Creates new form ListeCommandes
28      */
29
30     //System.out.println(commandes_json.length());
31     ArrayList<Commande> les_commandes = new ArrayList<>();
32 }
```

- **MyTableModel1.java** et **MyTableModel2.java** : ils permettent de personnaliser l'affichage des tableaux.

L'application utilise aussi une bibliothèque appelée **json-20231013.jar** pour lire les données envoyées depuis le site web, sous forme de fichier JSON. Par exemple, elle récupère la liste des commandes à traiter ou envoie une information pour dire qu'une commande a été acceptée ou refusée.

Grâce à RestoSwing, le restaurateur peut gérer les commandes facilement, directement depuis son ordinateur, sans passer par l'interface web. L'application fonctionne sur n'importe quel ordinateur avec Java installé (version 14 ou plus récente).