**AjouterCommande.fxml:**

<ComboBox fx:id="clientComboBox" prefWidth="200" />

**AjouterCommandeControllers:**

import org.example.services.ServiceClient;

@FXML

private ComboBox<String> clientComboBox;

**initialize:**

// Charger les clients depuis la base de données

try {

clients = new ServiceClient().getAllClients(); // Créer une instance de ServiceClient et appeler getAllClients()

ObservableList<String> observableClientNames = FXCollections.observableArrayList();

for (Client client : clients) {

String clientInfo = client.getId\_personne() + " - " + client.getNom\_personne() + " - " + client.getPrenom\_personne();

observableClientNames.add(clientInfo);

}

clientComboBox.setItems(observableClientNames);

clientComboBox.getSelectionModel().selectFirst();

} catch (SQLException e) {

afficherAlerte(Alert.AlertType.ERROR, "Erreur", "Impossible de charger les clients : " + e.getMessage());

}

**ajouterCommande(javafx.event.ActionEvent event):**

// Vérifier si aucun client n'a été sélectionné

if (clientComboBox.getSelectionModel().isEmpty()) {

afficherAlerte(Alert.AlertType.ERROR, "Erreur", "Veuillez sélectionner un client.");

return;

}

// Obtenir l'ID du client sélectionné

String selectedClientInfo = clientComboBox.getSelectionModel().getSelectedItem();

int clientId = Integer.parseInt(selectedClientInfo.split(" - ")[0]);

int nombreArticle = selectedArticles.size();

Date delaisCommande = Date.from(LocalDate.now().plusDays(2).atStartOfDay(ZoneId.systemDefault()).toInstant());

// Créer une nouvelle commande avec les articles sélectionnés et le client sélectionné

commande = new Commande();

commande.setArticles(selectedArticles);

commande.setNombre\_Article(nombreArticle);

commande.setPrix\_Totale(prixTotal);

commande.setDelais\_Commande(delaisCommande);

commande.setId\_Personne(clientId);

**serviceClient :**

@Override

public List<Client> getAllClients() throws SQLException {

List<Client> clients = new ArrayList<>();

String query = "SELECT c.id\_personne, p.nom\_personne, p.prenom\_personne " +

"FROM client c " +

"INNER JOIN personne p ON c.id\_personne = p.id\_personne"; // Assurez-vous que les noms de vos tables et colonnes sont corrects

try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {

ResultSet resultSet = preparedStatement.executeQuery();

while (resultSet.next()) {

Client client = new Client();

client.setId\_personne(resultSet.getInt("id\_personne"));

client.setNom\_personne(resultSet.getString("nom\_personne"));

client.setPrenom\_personne(resultSet.getString("prenom\_personne"));

// Ajoutez d'autres propriétés de client si nécessaire

clients.add(client);

}

}

return clients;

}

**serviceCommande:**

@Override

public void ajouterCommande(Commande commande) throws SQLException {

if (commande.getNombre\_Article() < 1) {

throw new IllegalArgumentException("Une commande doit contenir au moins un article.");

}

// Requête pour insérer une nouvelle commande avec l'ID de la personne du client

String sql = "INSERT INTO commande (id\_personne, nombre\_article, prix\_totale, delais\_commande) VALUES (?, ?, ?, ?)";

PreparedStatement preparedStatement = connection.prepareStatement(sql, Statement.RETURN\_GENERATED\_KEYS);

// Insérer l'ID de la personne sélectionnée dans la commande

preparedStatement.setInt(1, commande.getId\_Personne()); // Supposons que vous avez une méthode getId\_Personne() dans la classe Commande pour récupérer l'ID de la personne

preparedStatement.setInt(2, commande.getNombre\_Article());

preparedStatement.setDouble(3, commande.getPrix\_Totale());

preparedStatement.setDate(4, new java.sql.Date(commande.getDelais\_Commande().getTime()));

preparedStatement.executeUpdate();

// Récupérer l'ID de la commande nouvellement insérée

ResultSet generatedKeys = preparedStatement.getGeneratedKeys();

int id\_Commande;

if (generatedKeys.next()) {

id\_Commande = generatedKeys.getInt(1);

} else {

throw new SQLException("Échec de la création de la commande, aucun ID généré.");

}

// Insérer les ID des articles associés à la commande dans la table de liaison commande\_article

for (Article article : commande.getArticles()) {

String insertCommandeArticleQuery = "INSERT INTO commande\_article (id\_commande, id\_article) VALUES (?, ?)";

PreparedStatement commandeArticleStatement = connection.prepareStatement(insertCommandeArticleQuery);

commandeArticleStatement.setInt(1, id\_Commande);

commandeArticleStatement.setInt(2, article.getId\_Article());

commandeArticleStatement.executeUpdate();

}

}

@Override

public List<Commande> afficherCommande() throws SQLException {

List<Commande> commandes = new ArrayList<>();

String sql = "SELECT c.id\_commande, c.nombre\_article, c.prix\_totale, c.delais\_commande, p.id\_personne, p.nom\_personne, p.prenom\_personne, GROUP\_CONCAT(a.id\_article) as id\_articles, GROUP\_CONCAT(a.nom\_article) as nom\_articles " +

"FROM commande c " +

"INNER JOIN commande\_article ca ON c.id\_commande = ca.id\_commande " +

"INNER JOIN article a ON ca.id\_article = a.id\_article " +

"INNER JOIN client cl ON c.id\_personne = cl.id\_personne " +

"INNER JOIN personne p ON cl.id\_personne = p.id\_personne " +

"GROUP BY c.id\_commande";

try (Statement statement = connection.createStatement();

ResultSet rs = statement.executeQuery(sql)) {

while (rs.next()) {

Commande commande = new Commande();

commande.setId\_Commande(rs.getInt("id\_commande"));

commande.setNombre\_Article(rs.getInt("nombre\_article"));

commande.setPrix\_Totale(rs.getDouble("prix\_totale"));

commande.setDelais\_Commande(rs.getDate("delais\_commande"));

commande.setId\_Personne(rs.getInt("id\_personne"));

commande.setNom\_Personne(rs.getString("nom\_personne"));

commande.setPrenom\_Personne(rs.getString("prenom\_personne"));

String[] idArticles = rs.getString("id\_articles").split(",");

String[] nomArticles = rs.getString("nom\_articles").split(",");

List<Article> articles = new ArrayList<>();

for (int i = 0; i < idArticles.length; i++) {

Article article = new Article();

article.setId\_Article(Integer.parseInt(idArticles[i]));

article.setNom\_Article(nomArticles[i]);

articles.add(article);

}

commande.setArticles(articles);

commandes.add(commande);

}

}

return commandes;

}

**commande :**

private int id\_Personne;

private String nom\_Personne;

private String prenom\_Personne;

AfficherCommandeControllers:

**generePDF:**

// Ajouter les informations du client

contentStream.showText("ID du client : " + commande.getId\_Personne());

contentStream.newLineAtOffset(0, -30);

contentStream.showText("Nom du client : " + commande.getNom\_Personne());

contentStream.newLineAtOffset(0, -30);

**createCardCommande:**

// Récupérer les détails de la personne associée à la commande

int idPersonne = commande.getId\_Personne(); // Supposons que vous avez une méthode getId\_Personne() dans la classe Commande pour récupérer l'ID de la personne

String nomPersonne = commande.getNom\_Personne();

String prenomPersonne = commande.getPrenom\_Personne();

Label idPersonneLabel = new Label("ID de la personne : " + idPersonne);

idPersonneLabel.setStyle("-fx-font-size: 14px;");

Label nomLabel = new Label("Nom de client : " + nomPersonne);

nomLabel.setStyle("-fx-font-size: 14px;");

// Création du code QR pour la commande

// Modification du texte pour inclure les informations du client

ImageView qrCodeImageView = generateQRCodeCommande("ID de la commande : " + commande.getId\_Commande() + " - ID du client : " + commande.getId\_Personne() + " - Nom du client : " + commande.getNom\_Personne() + " - Prix total : " + commande.getPrix\_Totale() + " - Date de livraison : " + commande.getDelais\_Commande() + " - Nombre d'articles : " + commande.getNombre\_Article(), 200, 200);

qrCodeImageView.setFitWidth(100);

qrCodeImageView.setFitHeight(100);

**base de donneés** : ajouter id\_personne et faire la relation entre commande et client