UNIVERSITÉ CHEIKH ANTA DIOP DE DAKAR

Projet : Applications de Gestion des Commandes

Nom : Ndeye Maguette Kane

Niveau : Master 1 SIR/Soir

Année académique : 2024 2025

Enseignant : Mr DEYE

Table des matières

I. Introduction Générale

Dans le cadre de ma formation en Master 1 Systèmes d'Information Répartis, j'ai réalisé deux projets de gestion des commandes. Le premier projet est une application desktop développée avec WPF en C#, et le second est une application web développée avec ASP.NET Core MVC. Les deux projets utilisent la base de données Northwind pour la gestion des commandes, produits et clients.

II. Application de Gestion WPF

2.1. Présentation

L'application WPF est une application de bureau permettant de gérer des commandes clients. Elle utilise une interface graphique moderne avec XAML et permet une interaction intuitive avec la base de données SQL Server (Northwind) incluant :

- L'ajout de nouvelles commandes avec plusieurs produits.
- L'affichage et la recherche de commandes existantes.
- La modification
- suppression de commandes.

2.2. Environnement de Développement

Langage : C#

Framework : .NET 8 (WPF)

IDE : Visual Studio 2022

Base de données : SQL Server (Northwind)

Accès aux données : ADO.NET avec Microsoft.Data.SqlClient

Frontend : WPF (XAML) Contrôles : DataGrid, TextBox, Button, DatePicker, StackPanel

Architecture : Séparation des responsabilités (fichiers : MainWindow, OrderService, modèles)

2.3. Architecture du projet

Architecture : Séparée en trois couches principales

- Interface utilisateur : MainWindow.xaml / MainWindow.xaml.cs

- Modèle : Order.cs, ProduitCommande.cs

- Service métier et accès aux données : OrderService.cs (utilise ADO.NET)

Logique de recherche, ajout, modification, suppression centralisée dans OrderService

Data-binding via ObservableCollection pour mise à jour dynamique de l'interface

2.4. Fonctionnalités principales

✳ Affichage de la liste des commandes.

Sur la page d'accueil de notre application nous avons la liste des commandes de notre Base avec L'orderID, l'OrderDate et le CustomerID.

Nous avons également deux barres de recherche pour le CustomerID et le OrderID

✳ Ajout d'une nouvelle commande avec sélection de produits et quantités.

- Une fenêtre dédiée permet la création d'une commande.
- L'utilisateur renseigne :
  - L'ID du client.
  - La date de la commande.
  - Une liste de produits avec quantité et prix.
- L'interface propose un DataGrid interactif :
  - Une colonne ComboBox affiche la liste des produits.
  - Le prix unitaire est automatiquement rempli après la sélection d'un produit.

✳ Modification et Suppression d'une commande existante.

- Les commandes existantes sont listées dans un DataGrid.
- L'utilisateur peut sélectionner une commande et :
  - La modifier.
  - La supprimer avec confirmation.

✳ Recherche par identifiant de client ou numéro de commande.

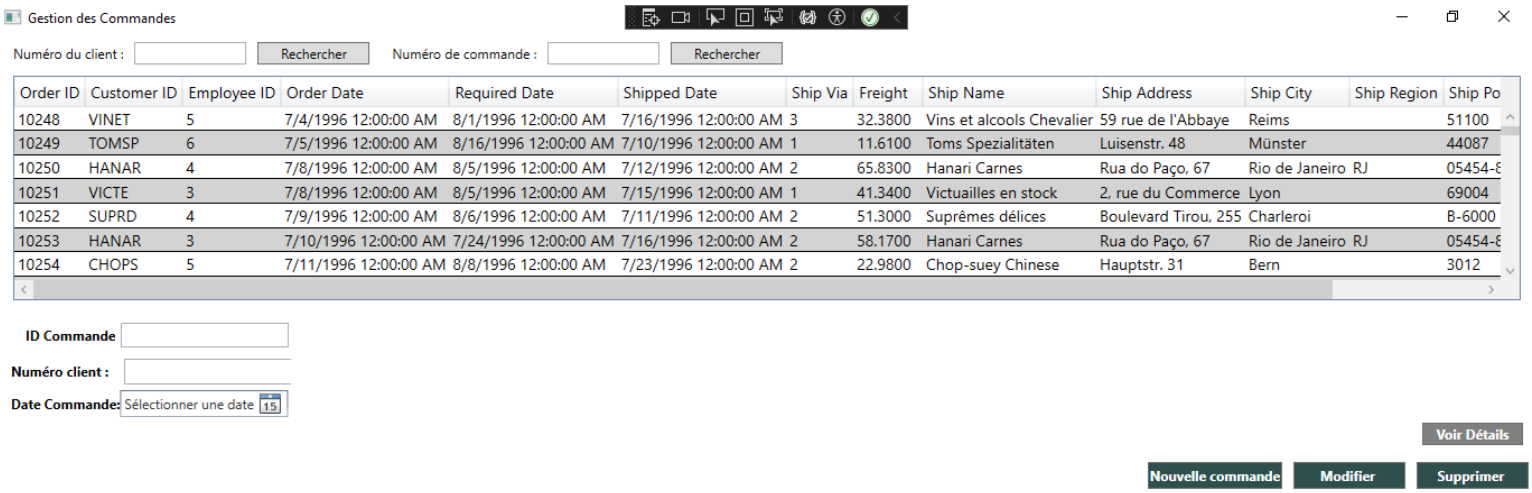
- Un champ de recherche permet de filtrer les commandes selon :
  - Le numéro de commande.
  - L'identifiant client.

2.5. Captures d'écran

◉ Page d'accueil

Cette capture montre la liste des commandes présentées dans un DataGrid.on a les colonnes de Orders

Image :

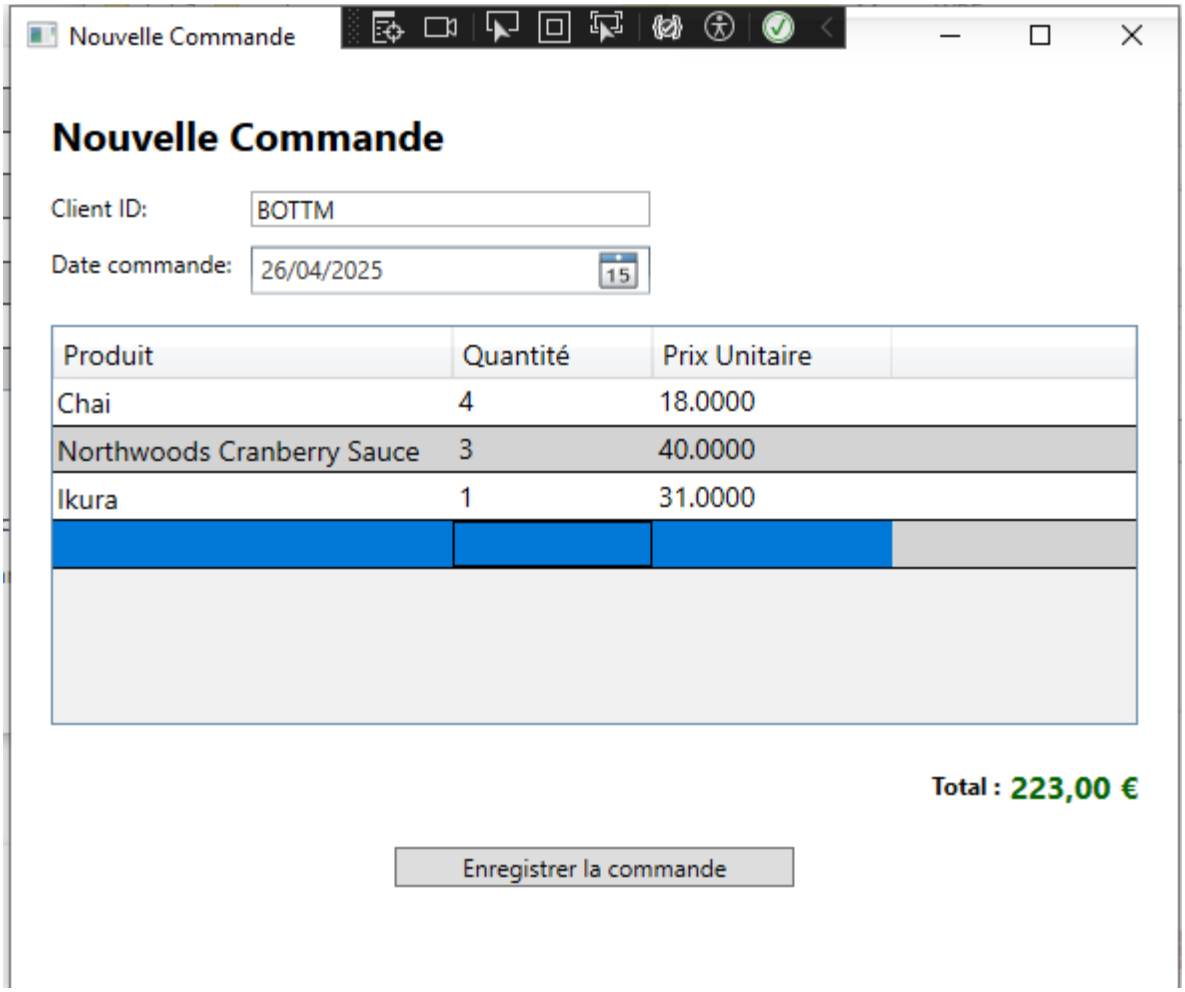


◉ Ajout d'une commande

Pour ajouter une commande il suffit de cliquer sur Nouvelle commande .

La fenêtre permet de sélectionner un client, une date et d'ajouter des produits. Le prix de chaque produit est directement exporté de la base de données, pas besoin de le saisir.

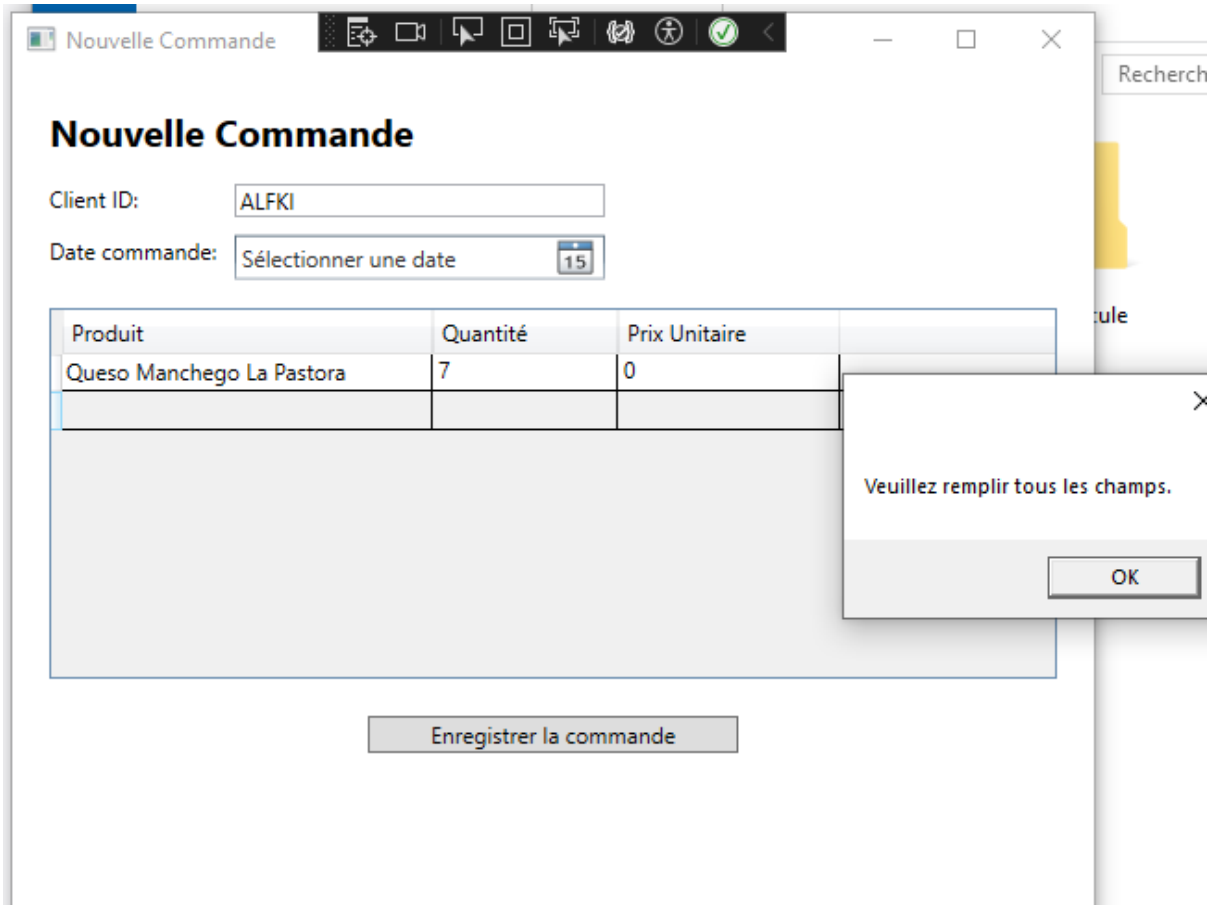
Image :

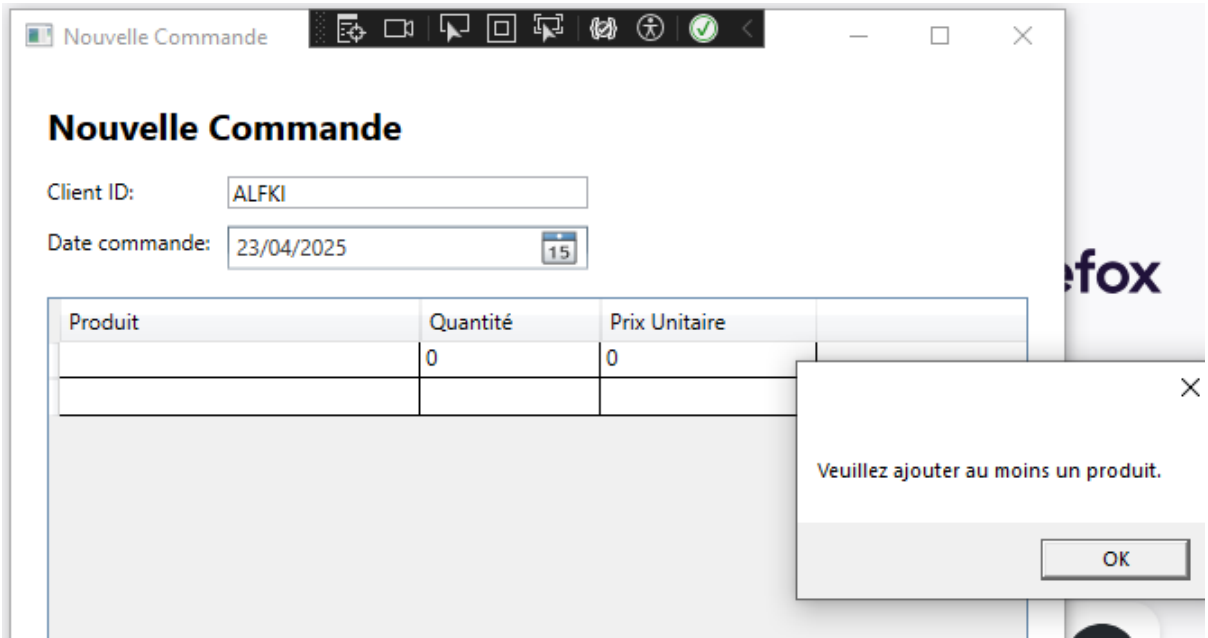


◉ Ajouter une commande sans mettre de produit

Lors de l'ajout d'une nouvelle commande, si l'utilisateur ne sélection aucun produit, il aura un message d'erreur

◉ Ajouter une commande sans remplir tous les champs



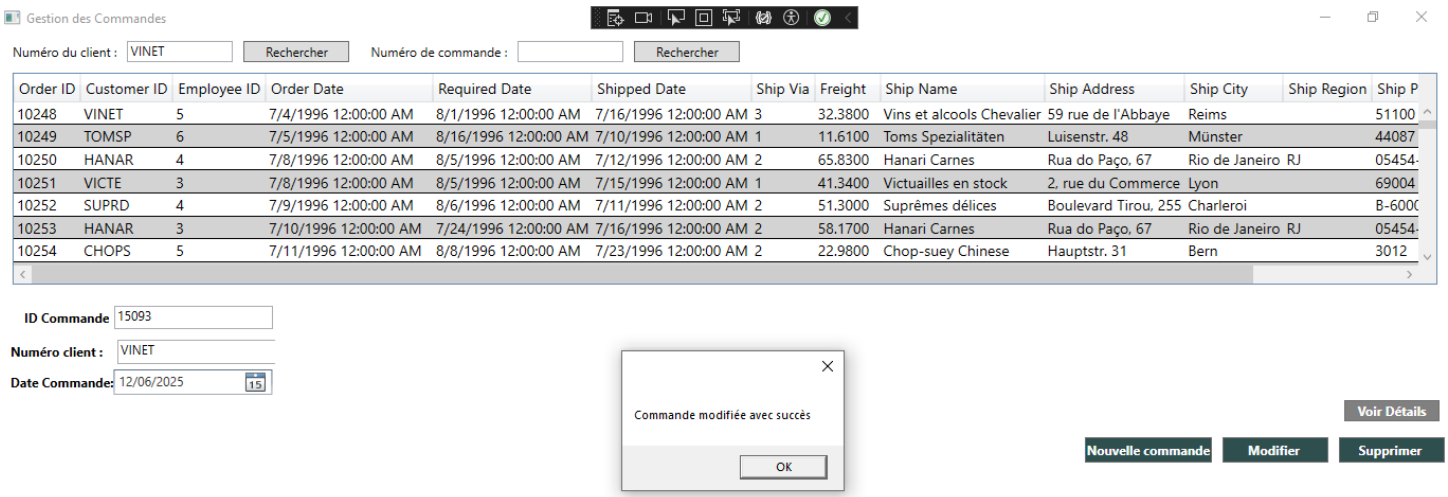


Et il en sera de même si tous les champs ne sont pas remplis.

◉ Modification d'une commande

Les champs sont pré-remplis et modifiables. Il suffit de cliquer sur la commande à modifier et les details apparaissent sur les champs. Ensuite il suffit de cliquer sur modifier la commande.

Image :



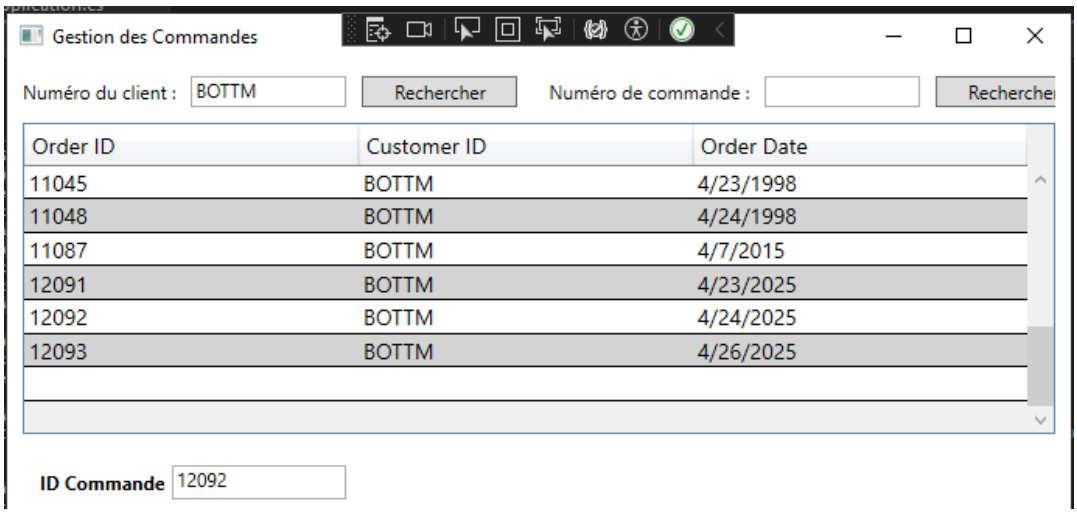
Verification de la modification:

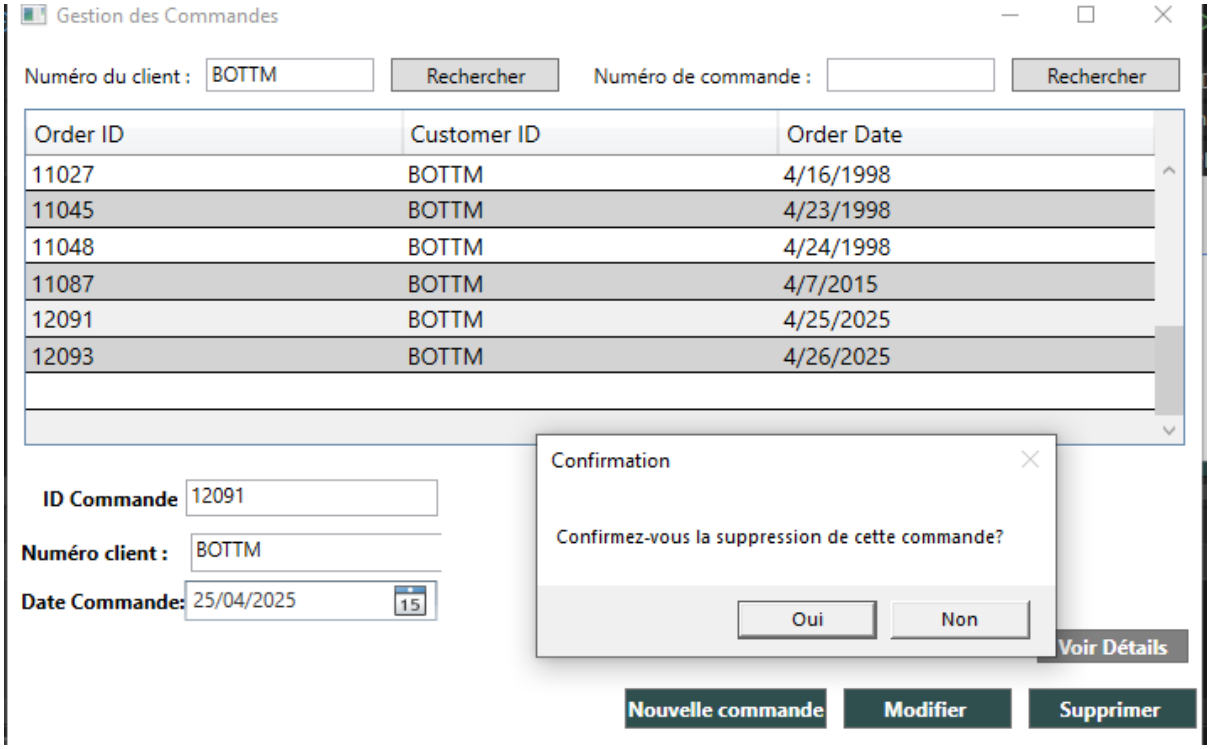
◉ Suppression d’une commande

Comme pour la modification ils suffit de cliquer sur la commande a supprimer, elle aparait sur les champs. Il sufffit ensuite de cliquer sur Supprimer .

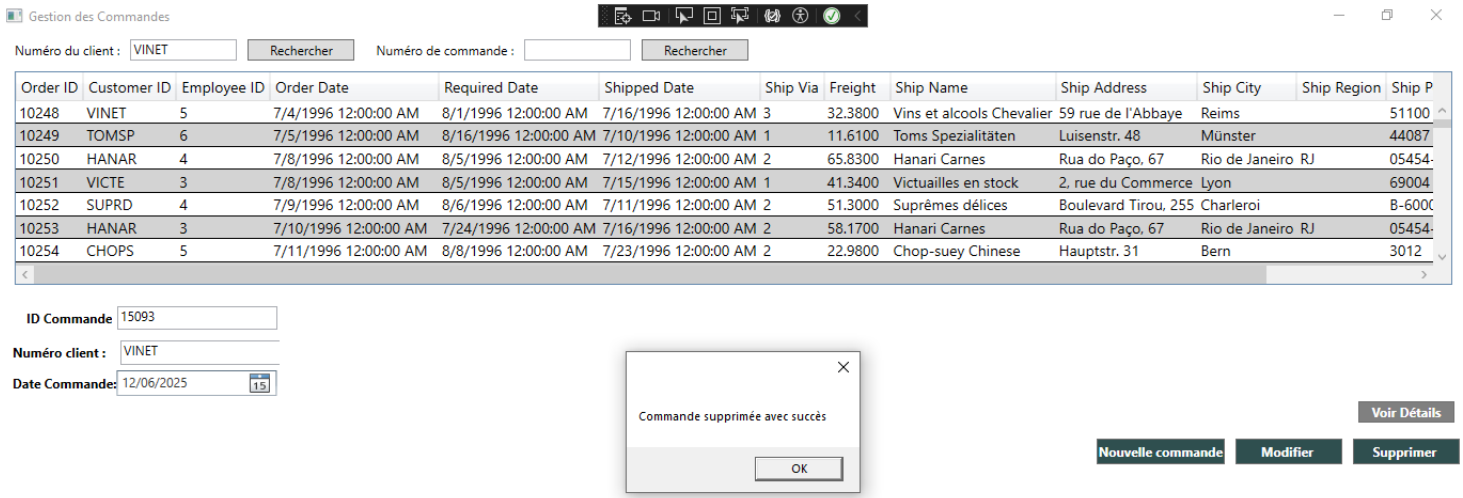
Une boîte de confirmation apparait pour verifier si on veut vraiment supprimer; si oui la commande est supprimée et sinon la requete est annulée. puis retour à la liste mise à jour.

Images :





Confirmer la suppression:

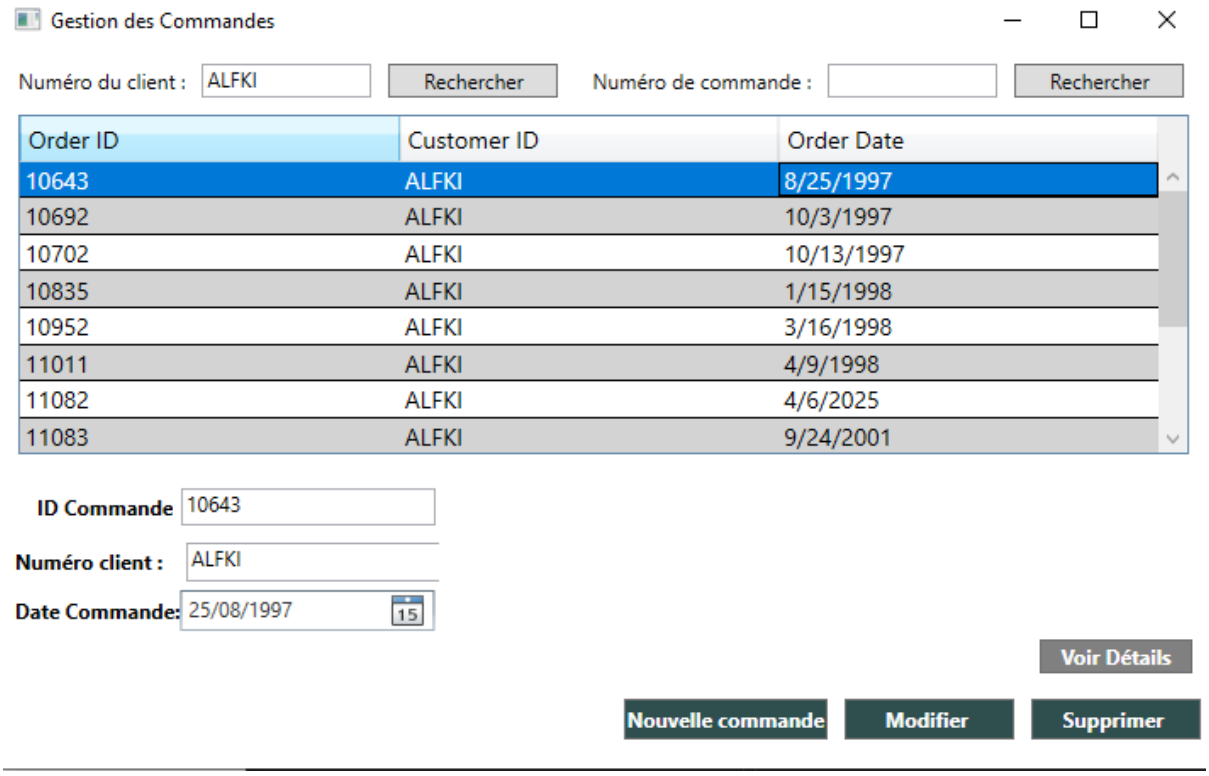


◉ Recherche par OrderID et CustomerID

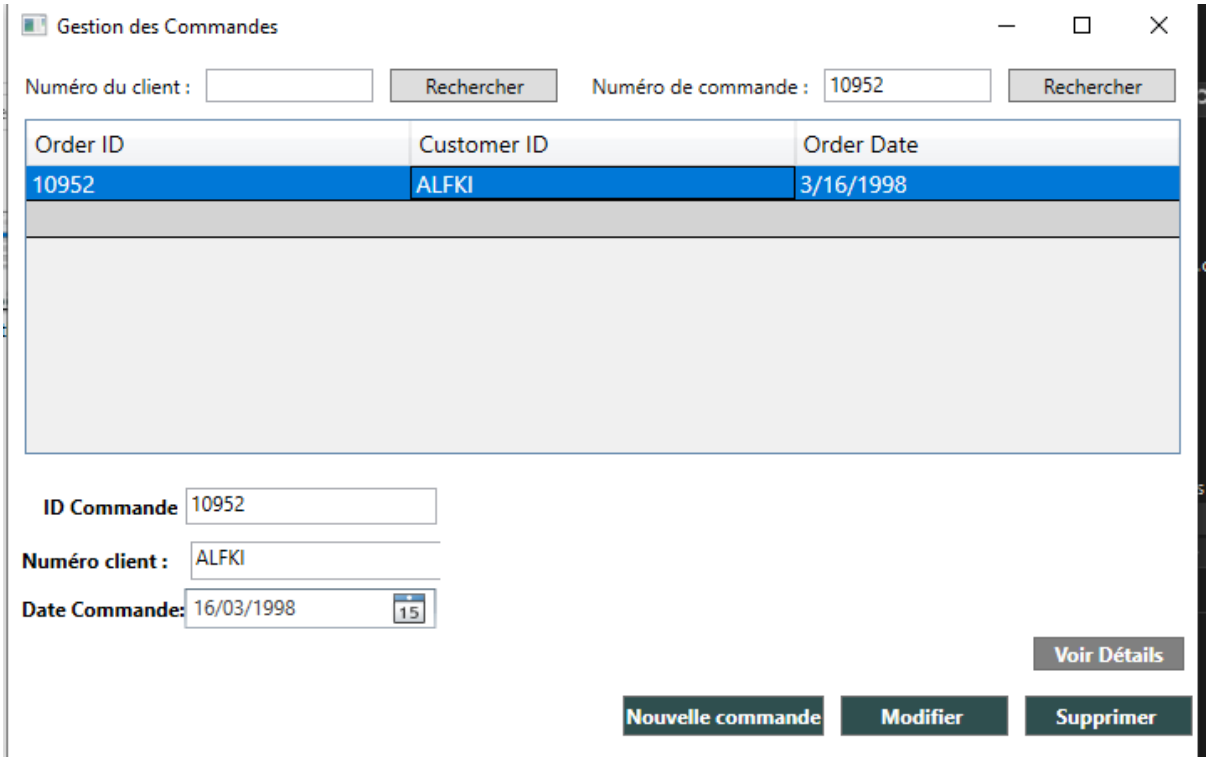
Filtrage dynamique par identifiant ou numéro de commande.

Images :

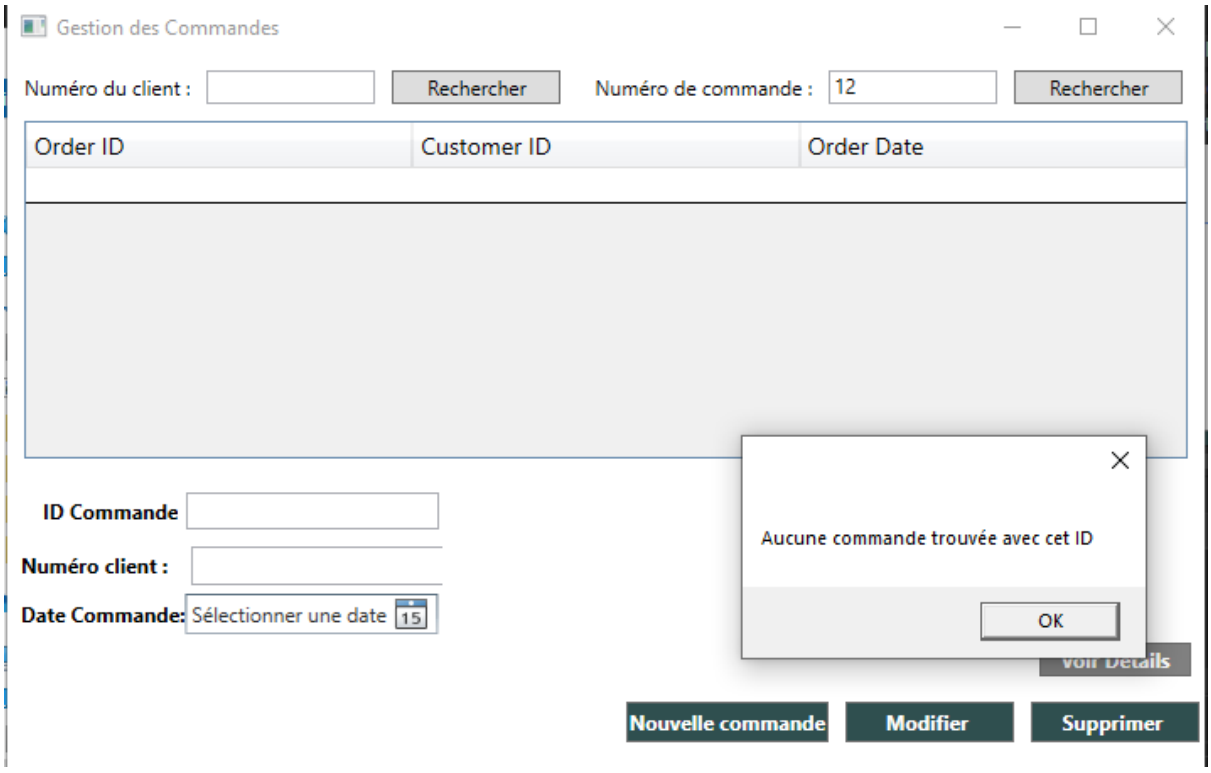
Par CustomerID



Par OrderID

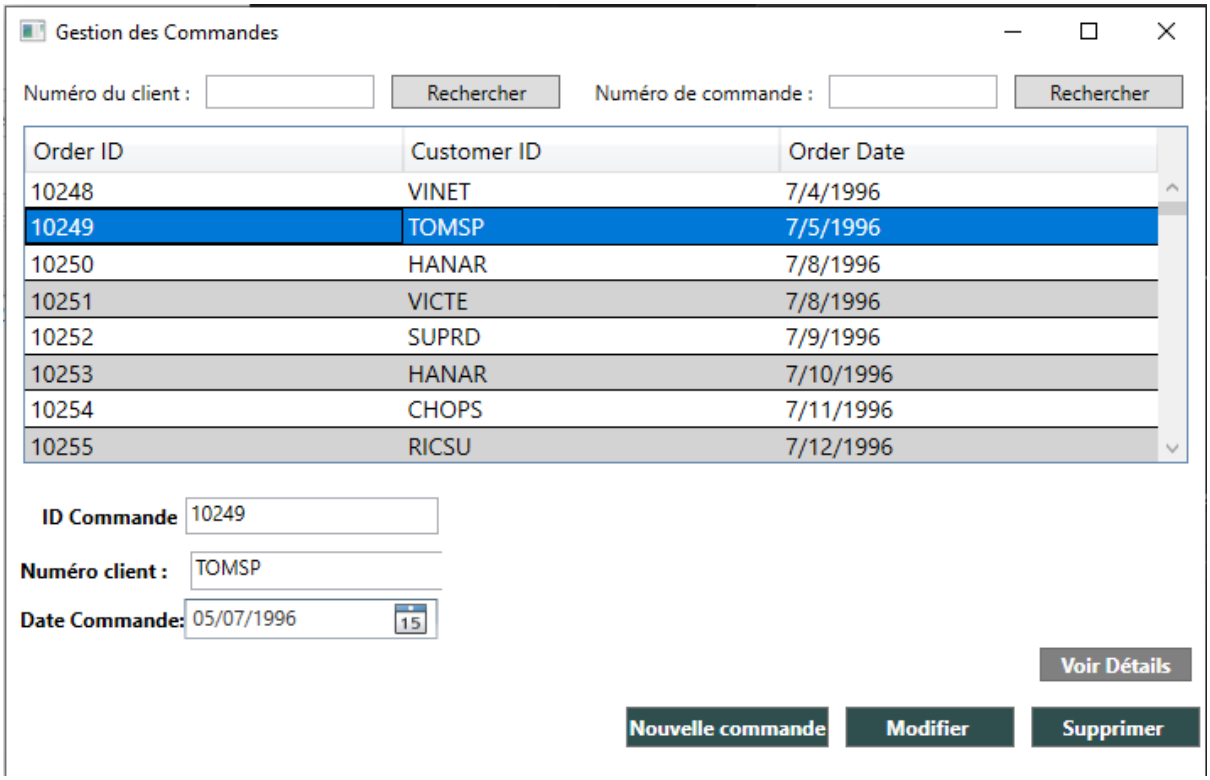


◉ Rechercher une OrderId qui n'existe pas



Au cas où l'utilisateur saisit un OrderId inexistant, on a un message d'erreur pour lui signaler qu'il n'y a aucune commande avec cet ID

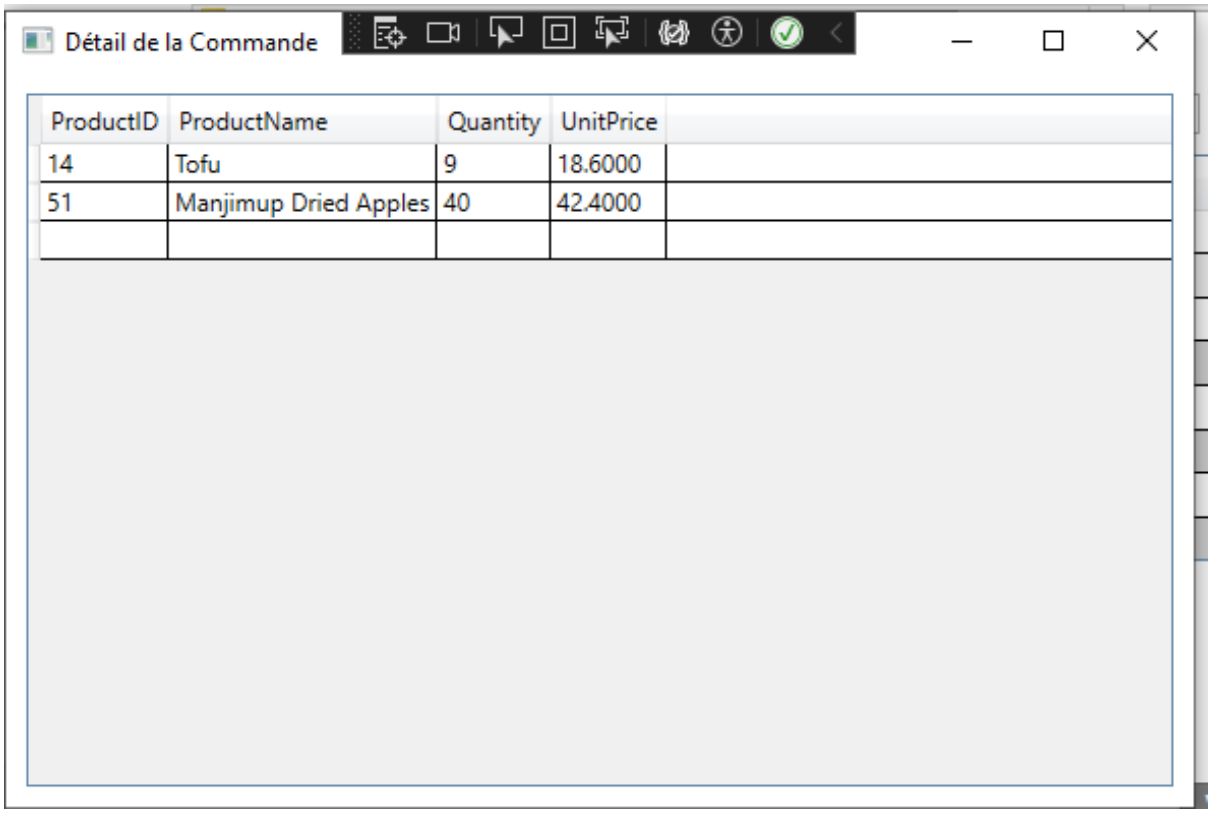
◉ Afficher une commande



Lors ce qu'on clique sur une commande dans le data grid les champs s'auto-remplissent de ses informations.

Afficher les Produits

Il suffit de cliquer sur **Voir details** pour afficher les produits liés a la commande sélectionnée.



2.6. Comment se passent les enregistrements

◉ Pour L'ajout d'une nouvelle commande

Il faut d'abord Récupérer les données saisies : L'utilisateur entre les informations de la commande (par exemple, CustomerID, OrderDate) dans les champs correspondants de l'interface.

Ensuite vient la création de la requête SQL : Une requête INSERT INTO est préparée pour insérer les nouvelles données dans la table Orders.

Puis on passe a l'éxécution de la requête : La requête est exécutée via une connexion SQL ouverte, insérant ainsi la nouvelle commande dans la base de données.

Et Enfin on actualise l'interface : Le DataGrid est mis à jour pour afficher la nouvelle commande.

◉ Pour La Modification

En premier lieu la sélectionne de la commande : L'utilisateur sélectionne une commande existante dans le DataGrid.

Ensuite on modifie des données : Les champs de l'interface sont remplis avec les données de la commande sélectionnée, que l'utilisateur peut modifier.

Puis on procède a la création de la requête SQL : Une requête UPDATE est préparée pour mettre à jour les données de la commande dans la base de données.

Apres cela on Exécute de la requête : La requête est exécutée, mettant à jour les informations de la commande.

Et enfin on Actualise l'interface : Le DataGrid est rafraîchi pour refléter les modifications.

◉ Pour La Suppression

D'abord on sélectionne la commande : L'utilisateur sélectionne la commande à supprimer dans le DataGrid.

Ensuite il faut Confirmer la suppression : Une confirmation peut être demandée à l'utilisateur pour éviter les suppressions accidentelles.

Puis on procède a la sréation de la requête SQL : Une requête DELETE est préparée pour supprimer la commande de la base de données.

Apres cela on passe a l'exécution de la requête : La requête est exécutée, supprimant la commande.

Et Enfin on actualise l'interface : Le DataGrid est mis à jour pour ne plus afficher la commande supprimée.

III. Application ASP.NET Core MVC

3.1. Présentation

Cette application web permet de gérer les commandes via une interface responsive conçue avec Bootstrap. Le modèle MVC est respecté, avec une séparation claire entre les modèles, les vues et les contrôleurs.

3.2. Environnement de Développement

- Langage : C#
- Framework : ASP.NET Core MVC
- IDE : Visual Studio 2022
- Base de données : SQL Server (Northwind)
- ORM : Entity Framework Core
- Frontend : HTML, CSS, Bootstrap 5

3.3.Architecture du projet

Le projet suit l'architecture MVC :

- Models : représentent les entités de la base de données.
- Views : fichiers .cshtml pour afficher l'interface utilisateur.
- Controllers : classes C# qui contrôlent le flux entre les vues et les modèles.

3.4. Fonctionnalités principales

✳ Visualisation des commandes avec pagination:

La page principale affiche toutes les commandes sous forme de tableau avec :

-Pagination personnalisée

-Barre de recherche par numéro de commande ou ID client

✳ Ajout d'une commande avec formulaire.

La création permet de :

- Saisir un client, une date
- Ajouter des produits avec leur quantité et prix unitaire.

La saisie utilise une interface avec DataGrid ou formulaire dynamique.

✳ Modification d'une commande existante.

Cette section Permet d'éditer une commande existante. Les listes déroulantes sont pré-remplies grâce à ViewBag.

✳ Suppression avec confirmation.

Avant suppression, une page de confirmation est affichée.
Si des Order_Details sont liés, une suppression en cascade est effectuée via le code suivant :

```
var orderDetails = _context.OrderDetails.Where(od => od.OrderID == id);

_context.OrderDetails.RemoveRange(orderDetails);

_context.Orders.Remove(order);

await _context.SaveChangesAsync();
```

✳ Affichage du détail d'une commande.

Affiche les informations complètes d'une commande sélectionnée :

- Détails client, date
- Liste des produits commandés, quantités, prix

✳ Recherche par client ou identifiant de commande.

On peut filtrer nos données par OrderID et CustomerID

✳ Interface utilisateur et Bootstrap

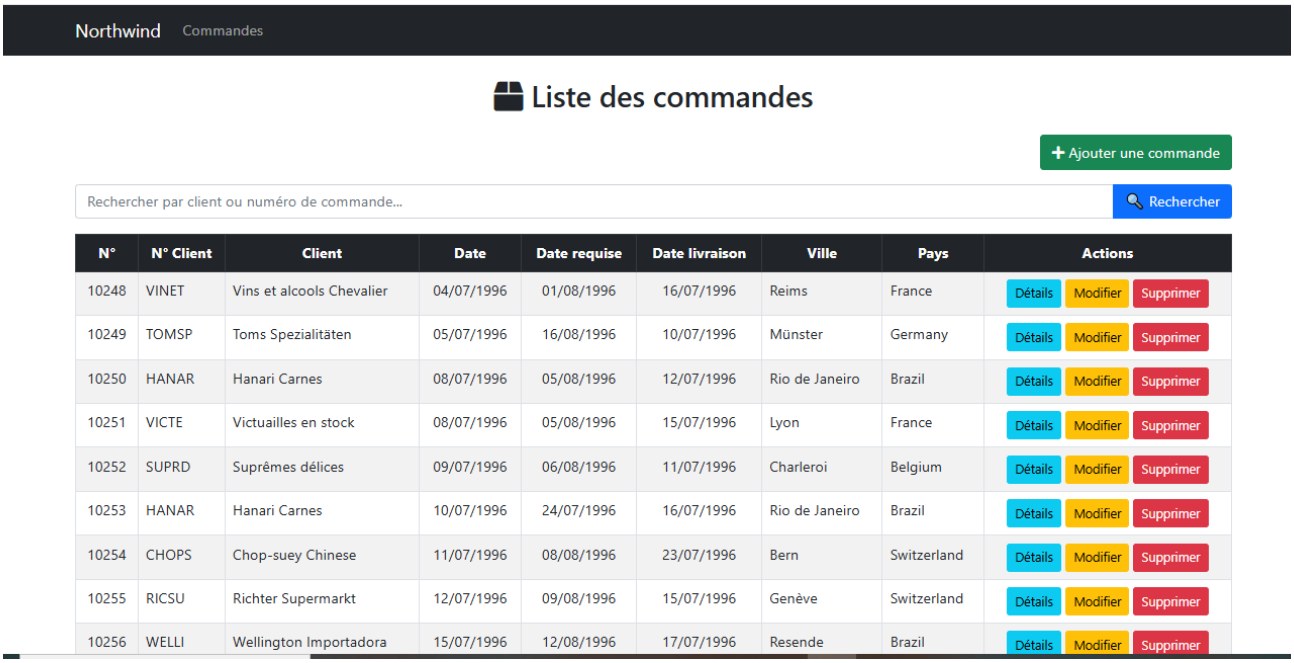
L'interface est construite avec Bootstrap 5, ce qui permet :

- Une compatibilité mobile (responsive design)
- Des boutons et tableaux élégants
- Des formulaires bien structurés
- Utilisation des classes Bootstrap : .container, .btn, .form-control, .table, etc.

3.5. Captures d'écran

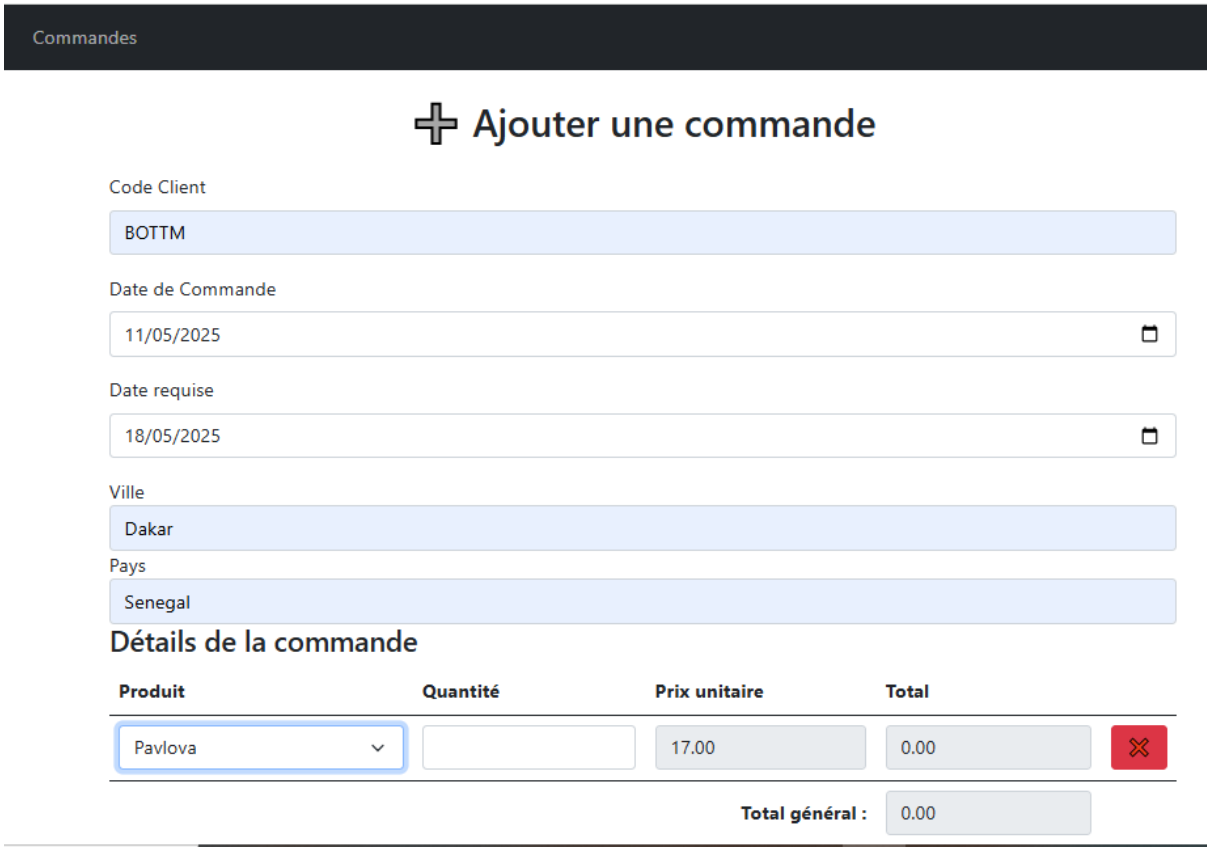
◉ Page d'accueil

Dans L'ecran d'accueil nous avons une barre de recherche, les information des commandes(OrderID, CustomerID, OrderDate), les actions par rapport à ces commandes, et un bouton d'ajout de commande

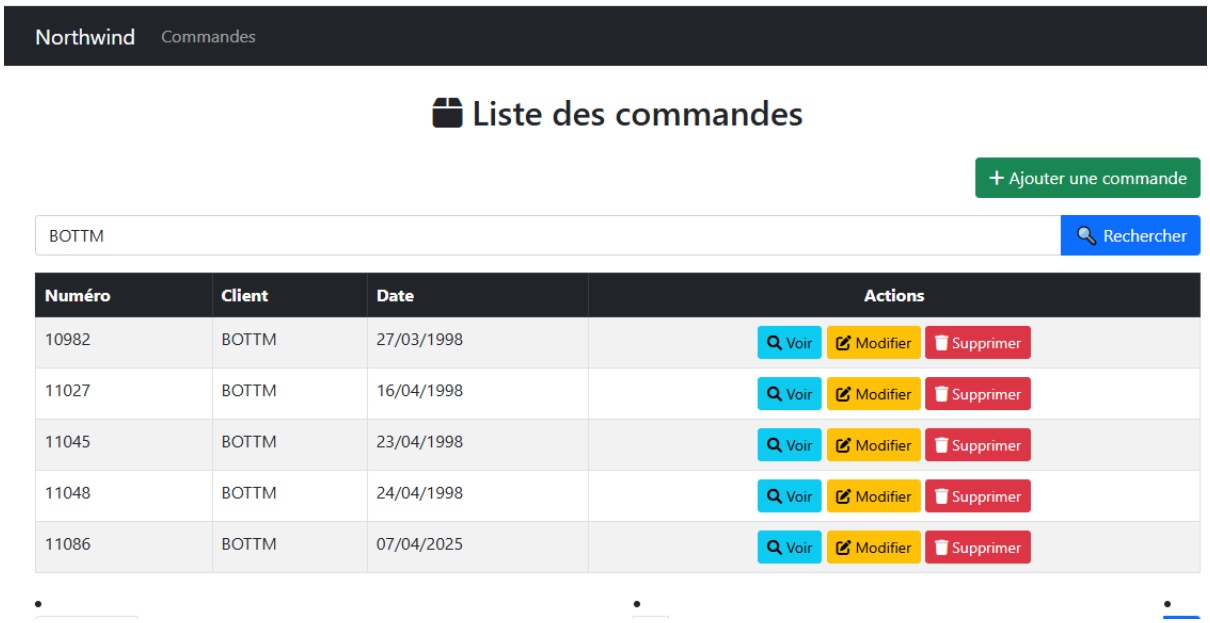


◉ Ajout d'une commande

Formulaire pour ajouter une commande à un client.



Apres l'ajout on peut voir la nouvelle commande ajoutée:



◉ Modification d'une commande

Nous avons un Formulaire avec pré-remplissage et bouton de validation. Nous avons la possibilité de modifier les détails de la commande

Apres modification nous pouvons voir le resultat .

◉ Suppression d'une commande

Sur cette page on peut visualiser les informations de la commande avant de tout supprimer

Image :

Supprimer cette commande ?

Numéro Commande : 10248

Client : VINET

Date : 04/07/1996

Produits de la commande :

Produit	Quantité	Prix unitaire
Queso Cabrales	12	14,00 €
Singaporean Hokkien Fried Mee	10	9,80 €
Mozzarella di Giovanni	5	34,80 €

Supprimer Annuler

◉ Détail d'une commande

Affichage détaillé du client, produits, Quantité, Prix Unitaire et prix.. Nous Avons aussi la date et le OrderID

Image :

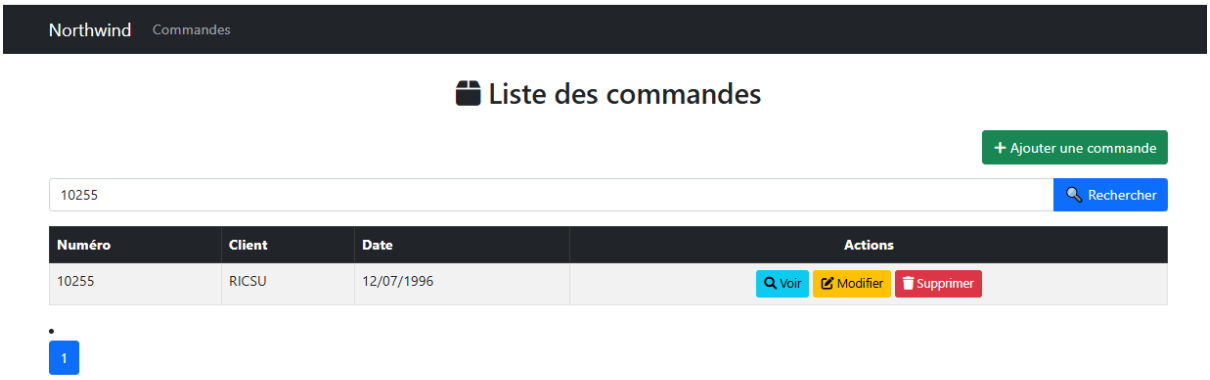
◉ Recherche

Champ de recherche et affichage des résultats.

Image :

Par CustomerID

Par OrderID:



3.6. Comment se passent les enregistrements

◉ Pour L'ajout d'une nouvelle commande

Lorsqu'un utilisateur remplit le formulaire d'ajout et clique sur "Enregistrer", les données du formulaire sont envoyées (via HTTP POST) à la méthode Create du contrôleur. Cette méthode reçoit un objet (OrderCreateViewModel), qu'elle utilise

pour construire une entité Order et ses OrderDetails. Le contrôleur ajoute ensuite cette entité au contexte de base de données (_context.Orders.Add(order)) et appelle SaveChanges() pour enregistrer les données dans la base. Une fois terminé, l'utilisateur est redirigé vers la liste des commandes.

◉ Pour La Modification

Lors de la modification, les données existantes sont d'abord affichées dans un formulaire pré-rempli. L'utilisateur modifie ce qu'il souhaite puis clique sur "Valider". Les nouvelles valeurs sont envoyées au serveur, qui récupère l'entité originale à partir de la base avec son ID, elle met à jour ses propriétés, et appelle SaveChanges(). Si la commande contient aussi des détails (comme des produits), ces derniers peuvent être mis à jour, supprimés ou ajoutés selon le formulaire. Tout est ensuite sauvegardé dans la base.

◉ Pour La Suppression

Quand l'utilisateur clique sur "Supprimer", le contrôleur appelle une méthode qui récupère l'entité concernée à partir de son ID. Cette entité est ensuite supprimée du contexte de base de données avec _context.Orders.Remove(order), puis SaveChanges() est appelé pour valider la suppression dans la base. L'utilisateur est ensuite redirigé vers la vue d'index ou reçoit une confirmation.

3.7. Problèmes rencontrés et solutions

1. Conflit lors de la suppression d'une commande

Problème: Erreur de contrainte étrangère

Solution: Suppression manuelle des détails avant la commande

2. Perte de recherche après pagination :

Problème: Les pages suivantes montraient toutes les commandes

Solution: Persistance du searchString dans ViewData et dans la pagination

3. Affichage des listes déroulante dans la vue EDITe :

Problème: Les pages suivantes montraient toutes les commandes

Solution: Persistance du searchString dans ViewData et dans la pagination

4. L'ID était bloqué à l'ajout d'une commande

Problème: Le champ OrderID était saisi manuellement alors qu'il est auto-incrémenté (IDENTITY)

Solution: Suppression du champ dans le formulaire (l'ID est géré automatiquement par SQL Server).

5. Les détails de commande ne s'affichaient pas

Problème: Les données n'étaient pas chargées via Include()

Solution: Utilisation de Include(o => o.OrderDetails).ThenInclude(od => od.Product) dans le contrôleur.

6. Incohérences entre types C# et SQL

Problème: Erreur InvalidCastException sur le champ Quantity. Quantity est smallint en SQL Server, mais on l'avait défini en int.

Solution: Correction dans le modèle C# : passage à short.

IV. Conclusion

Ces deux projets ont été une expérience enrichissante. Le projet WPF m'a permis de me familiariser avec les interfaces desktop et la gestion des données via SQL Server. Le projet ASP.NET m'a apporté une compréhension approfondie du développement web, de l'architecture MVC, ainsi que de l'utilisation d'Entity Framework Core. Il est vrai que j'ai rencontré beaucoup de problèmes et j'ai dû faire énormément de recherches pour finaliser ce projet mais ce fut très instructif et motivant. Ces compétences seront déterminantes pour mes futurs projets professionnels.