

**Nom de l’entreprise : Valkorion.h**

**Nom du projet : Nebula**

**Adresse : 204 Boulevard Godard**

**Tel : 0606060669**

**Email : contact@valkorion-h.fr**

|  |
| --- |
|  |

DOCUMENTATION CLIENT

Table des matières

[I. DASHBOARD : 2](#_Toc121214980)

[II. GESTION DES EMPLOYÉS : 3](#_Toc121214981)

[Ajout d’employés : 3](#_Toc121214982)

[Edition d’employés : 4](#_Toc121214983)

[III. GESTION DES CLIENTS : 4](#_Toc121214984)

[Ajout de client : 5](#_Toc121214985)

[IV. GESTION DES STOCK : 6](#_Toc121214986)

[Ajout de stock : 6](#_Toc121214987)

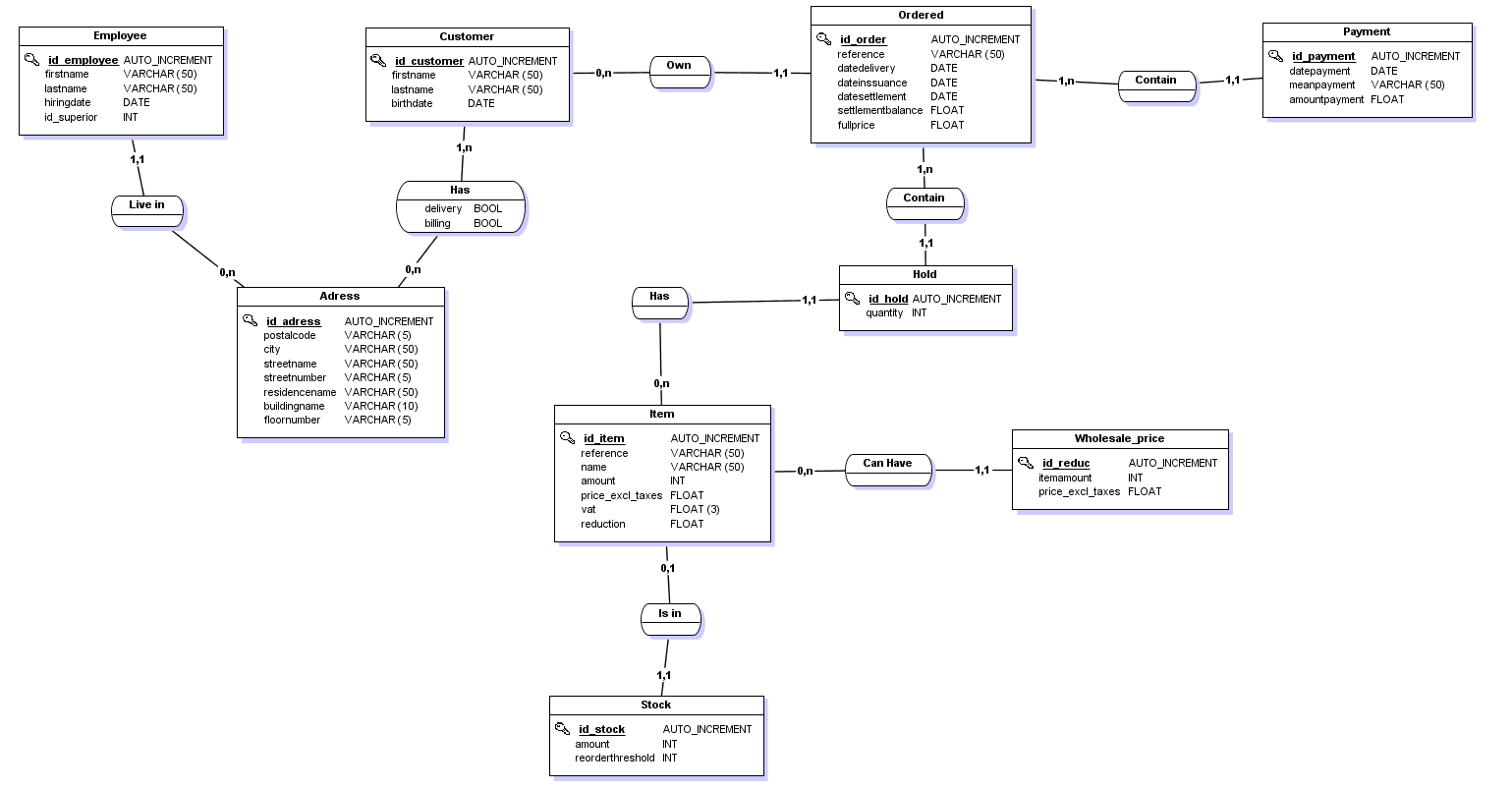
[V. GESTION DES COMMANDES : 6](#_Toc121214988)

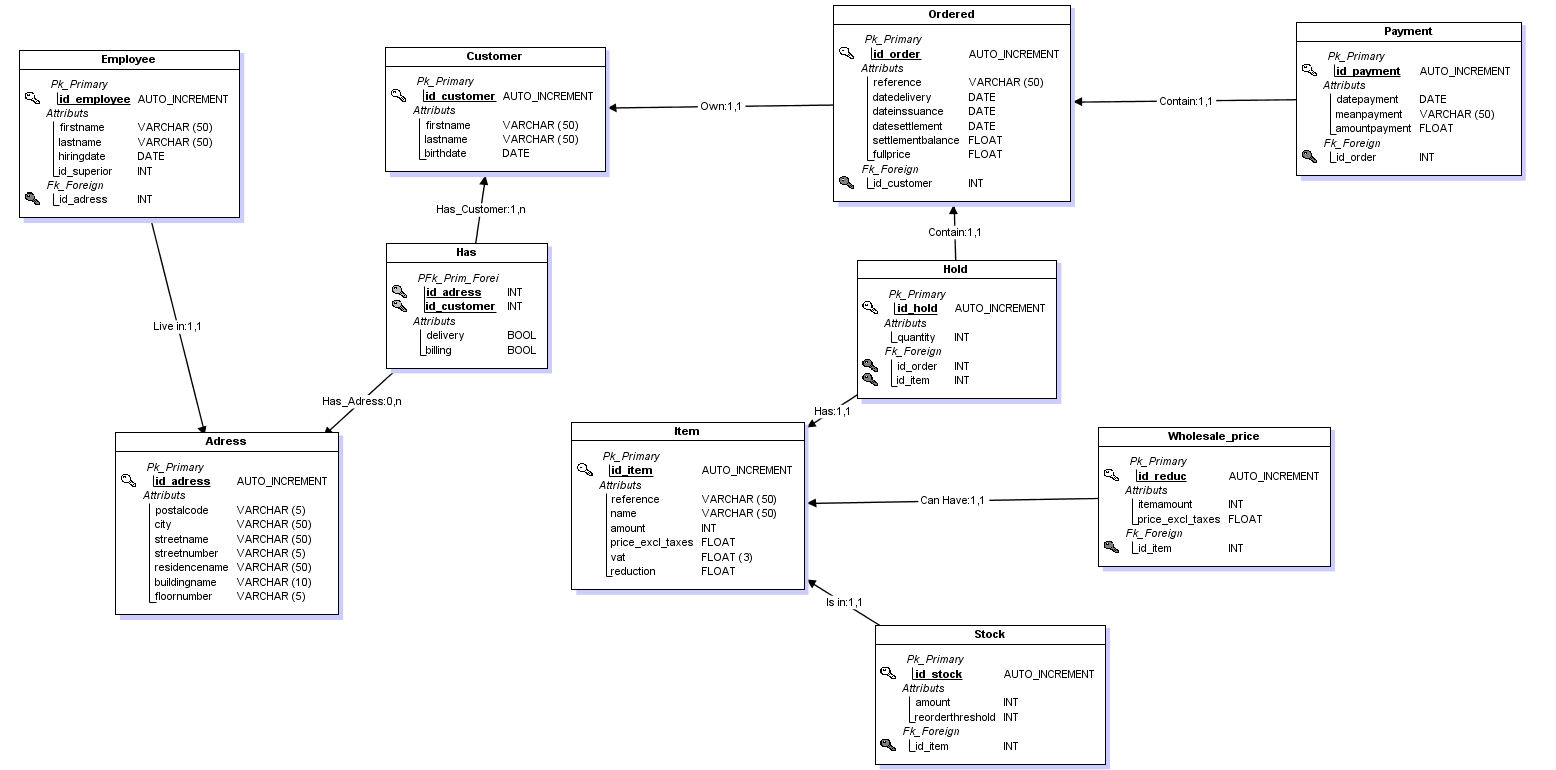
[Edition d’une commande : 7](#_Toc121214989)

# BASE DE DONNÉE :

La base de données que nous utilisons est une base de données SQL Server. ### Cette dernière est hébergé sur un rasberry pi et est disponible via l’ip : .###

## MCD & MLD :

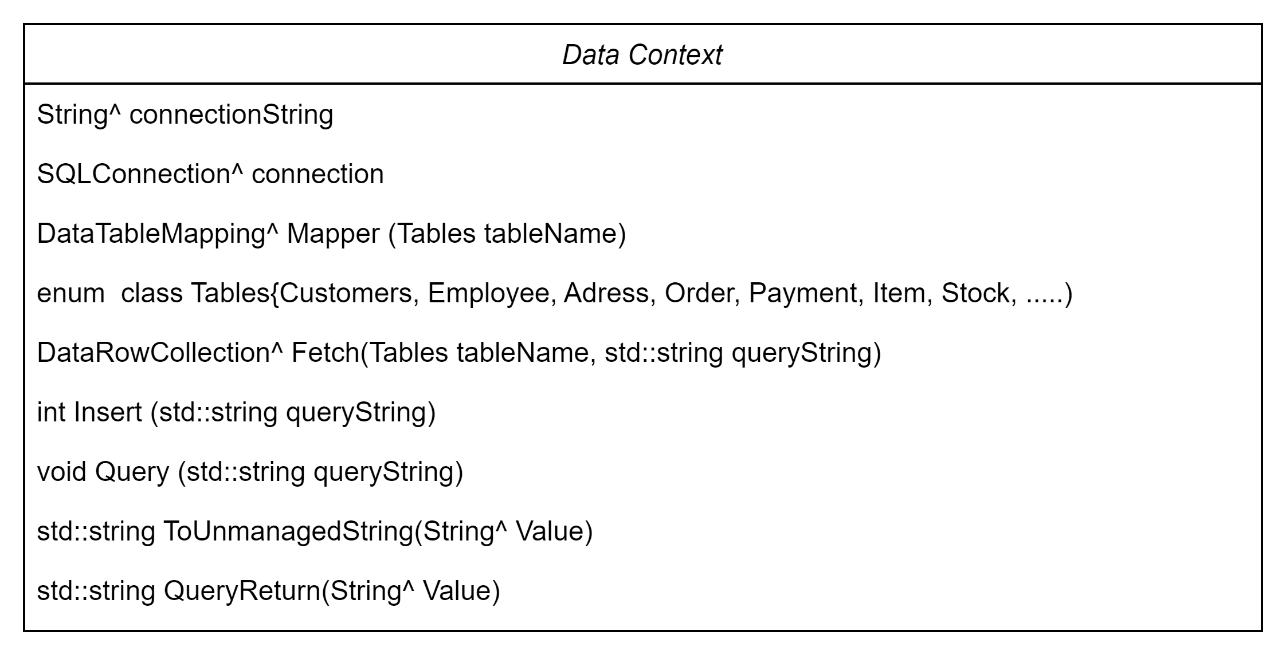




Nous avons créer notre base de donnée en respectant la forme normale.  
Nous stockons dans la table « Hold » la quantité d’un article commandé afin d’éviter l’apparition de plusieurs ligne.  
Il sera simple de créer de nouvelle colonne dans une table, faites juste attention au redondance.

# COMMUNICATION CODE & BDD :

## Data Context :



Voici le diagramme de classe de « Data Context ». Cette classe permet la liaison directe avec notre BDD.  
Elle contient la « connectionString » qui sert a définir à quel BDD le logiciel se connectera, différentes requête classique (« Fetch », « Insert », « Query », « QueryReturn ») qui seront appelé dans nos différents contrôleurs nommés « Service ». Cette classe contient également Mapper permettant de réorganiser les données obtenus via les différentes requetés, ainsi qu’une méthode « ToUnmanagedString » permettant de convertir les string managé en string non managé. Cette méthode est effectivement utile car notre programme est entièrement managé afin que la gestion de la mémoire soit optimale.

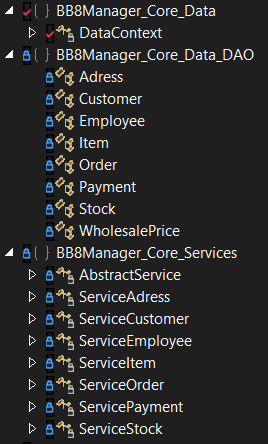
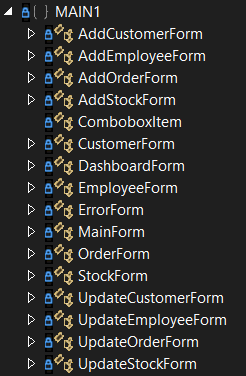
## Contrôleur :



Voici le diagramme de classe de notre contrôleur d’adresse. Celui-ci permet d’effectuer des requêtes en rapport avec la table « Adress ».  
Ce contrôleur contient les méthodes classiques. Certains contrôleurs peuvent en contenir plus selon le besoin. Il sera donc simple pour vous de créer des méthodes, en effet chaque contrôleur appelle des méthodes de la classe « DataContext » et avec les méthodes déjà présentes dans « DataContext » il vous sera facilement possible de créer de nouvelle requête dans votre contrôleur.

# LE CODE DU LOGICIEL:

## Arborescence :

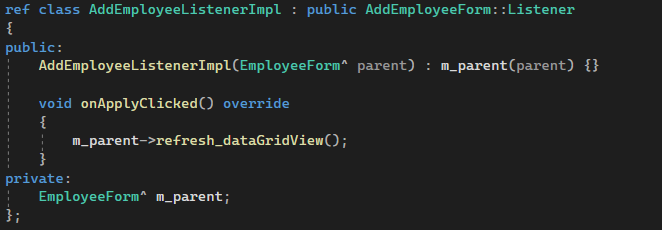
« BB8Manager\_Core\_Data\_DAO » contient nos différents modèles.  
« BB8Manager\_Core\_Services » contient nos différents contrôleurs.  
« MAIN1 » contient nos différents formulaires.

## Formulaire :

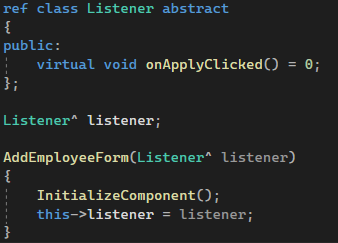
Il existe 4 types de formulaire :

* Le form principal « MainForm » qui contient le menu déroulant ainsi que l’emplacement des différents panel à afficher.
* Le form panel (par exemple « EmployeeForm ») qui vient se fixer sur le main dans l’emplacement adéquat
* Le form d’ajout/edition qui s’ouvre lorsque l’utilisateur souhaitera, comme son nom l‘indique, ajouter ou éditer.
* Le form d’erreur qui s’ouvre lorsque l’utilisateur effectue mal une action. Un message d’erreur est joint sur ce dernier.

Concernant les formulaires d’ajout et d’édition nommé « AddXXXForm » et « UpdateXXXForm » ou XXX Correspond au nom du modele par exemple « AddCustomerForm », il était nécessaire de créer une classe listener afin de permettre la communication entre le Form pricipale et le Form d’ajout ou de modification.   
Par exemple notre « EmployeeForm » contient :



Et notre « AddEmployeeForm » contient :



Cela nous permet donc d’appeler une méthode, en l’occurrence « refresh\_dataGridView » qui va nous permettre d’actualiser les données de la datagridview lorsqu’un utilisateur sera ajouté.

## Modèle :

Une image contenant texte

Description générée automatiquement

Un modèle est constitué d’attribut correspondant aux colonnes de la table auquel il réfère dans la BDD cela comprend aussi bien les primary key que les foreign key. Notre classe contient également un constructeur surchargé ainsi que des Setter et Getter pour chaque attribut.  
Si vous souhaitez donc ajouter des tables par la suite il faudra donc penser à créer un modèle correspondant.