

**Rapport de Projet – RAD**

Gestion de la vente de vin

Table des matières

[1. Introduction 4](#_Toc472709356)

[1.1. Description du projet 4](#_Toc472709357)

[1.2. Description de l’application 4](#_Toc472709358)

[1.3. Objectifs 4](#_Toc472709359)

[2. Respect des Contraintes 4](#_Toc472709360)

[2.1. Types de données 4](#_Toc472709361)

[2.2. Association 1.n 5](#_Toc472709362)

[2.3. Association n.n 5](#_Toc472709363)

[2.4. Association réflexive 5](#_Toc472709364)

[2.5. Séquences, triggers et contraintes 5](#_Toc472709365)

[3. Application 5](#_Toc472709366)

[3.1. Connexion 5](#_Toc472709367)

[3.1.1. Sécurité 6](#_Toc472709368)

[3.1.2. Modèles d’autorisation 6](#_Toc472709369)

[3.2. Modèles de données 7](#_Toc472709370)

[3.2.1. Explications 7](#_Toc472709371)

[3.2.2. Modèle conceptuel de données / Logical Model 9](#_Toc472709372)

[3.2.3. Modèle logique de données / Relational Model 10](#_Toc472709373)

[3.3. Journalisation 11](#_Toc472709374)

[3.4. Fonctionnalités choisies 11](#_Toc472709375)

[3.5. Meilleure fonctionnalité 11](#_Toc472709376)

[3.6. Problèmes rencontrés / Solutions trouvées 12](#_Toc472709377)

[3.6.1. Nommage de champ Date 12](#_Toc472709378)

[3.6.2. Modification du modèle relationnel 13](#_Toc472709379)

[3.6.3. Tablespace plein 13](#_Toc472709380)

[4. Conclusion 13](#_Toc472709381)

[5. Références 14](#_Toc472709382)

# Introduction

## Description du projet

Notre projet est une application de gestion pour la vente de vins, *Gesto’Vin*. Notre entreprise se propose en tant que revendeur pour les particuliers produisant leur propre vin. Nous agissons donc en tant qu’intermédiaire entre ceux-ci et les clients.

## Description de l’application

*Gesto’Vin* comprend tout d’abord la gestion des utilisateurs, c’est-à-dire les collaborateurs qui auront accès à l’application, car ce sont eux qui entreront les commandes des clients dans le système. Ils ont bien évidemment accès aux fonctions d’ajout, de modification, de suppression et de visualisation du contenu.

Les formulaires et états sont créés en fonction de nos tables et associations (modèle de données).

Un graphique permet aux utilisateurs de visualiser les ventes de vin par canton à des fins statistiques.

Nous utilisons également la journalisation des enregistrements afin de garder des traces sur certaines informations (les commandes des clients par exemple).

## Objectifs

Notre objectif pour cette application de gestion est d’entrer les commandes des clients dans le système afin que tout soit informatisé et centralisé.

Les utilisateurs pourront par la suite effectuer des statistiques afin de prendre différentes décisions quant aux ventes de vin.

# Respect des Contraintes

## Types de données

Les types de données les plus utilisés sont représentés. Voici quelques exemples d’utilisation :

* **NUMBER :** *Clés (primaires et secondaires), quantités, …*
* **VARCHAR2 :** *Noms, descriptions, …*
* **DATE :** *Dates des commandes*
* **LOB (BLOB ou CLOB) :** *Photos des vins*

## Association 1.n

Des associations 1.n sont présentes dans notre modèle. Par exemple, pour la liaison entre un fournisseur et un vin : ***un producteur produit 1 ou plusieurs vins***.

## Association n.n

Deux associations de type n.n sont également présentes au niveau du modèle de l’application. Elles se trouve au niveau des commandes de bouteilles par le client (***un client commande 1 ou plusieurs bouteilles. Une bouteille est commandée par 1 ou plusieurs clients***) et au niveau de la définition du prix en fonction du volume de la bouteille et du vin. Les tables associatives *Commande* et *Bouteille* sont donc créées.

## Association réflexive

Il est possible pour un fournisseur (particulier) d’avoir un représentant qui se charge du contact avec notre entreprise. Cela représentera une association réflexive au niveau de la table *Personne*.

## Séquences, triggers et contraintes

Chaque clé primaire est alimentée par une séquence. Cela est possible grâce à des triggers présents sur chaque table.

Les noms des contraintes respectent les standards définis dans la donnée du cas pratique.

# Application

## Connexion

Pour se connecter à l’application, nous utilisons une table d’utilisateur qui se trouve dans la base de données. La table contient tous les utilisateurs qui pourrons, par la suite, se connecter à notre application de gestion. Ceux-ci ont un type et, en fonction de ce type, certains contenus sont masqués.

|  |  |  |
| --- | --- | --- |
| ***Username*** | ***Password*** | ***Type*** |
| ***murielle*** | *mumu16* | *ADMIN* |
| ***jean*** | *jeje16* | *COMMERCIAL* |
| ***paule*** | *papa16* | *VENDEUR* |

### Sécurité

Pour plus de sécurité, tous les mots de passe de la base de données sont hachés avant d’être stockés via le package ***API\_AUTHENTIFICATION*** présent dans notre base de données.

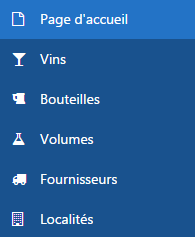
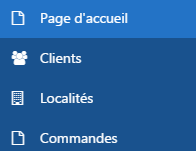
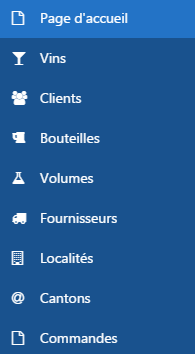
### Modèles d’autorisation

Afin de limiter le contenu de l’application uniquement aux personnes concernées, nous avons mis en place des modèles d’autorisation pour chacun des types d’utilisateurs présent dans notre base de données (étant donné la structure restreinte de notre entreprise, ces types ont été stockés dans une colonne de la table UTILISATEURS et non une table spécialement prévue pour cela).

Les modèles d’autorisations sont les suivants :

|  |  |  |
| --- | --- | --- |
| ***Type*** | ***Menu de navigation (onglets accessibles)*** | ***Description*** |
| ***ADMIN*** | *Bouteilles Cantons Clients Commandes Localités Fournisseurs Vins Volumes*  *(= tous)* | *L’administrateur a accès à toutes les données et fonctions afin de pouvoir gérer tous les problèmes et corriger les éventuelles erreurs commises par les autres collaborateurs.* |
| ***COMMERCIAL*** | *Bouteilles Localités Fournisseurs Vins Volumes* | *Le commercial s’occupe de tout ce qui concerne les relations avec les fournisseurs et leurs produits.* |
| ***VENDEUR*** | *Clients Commandes Localités* | *Le vendeur est chargé des relations avec les clients et des commandes que ceux-ci passent.* |

Avec ces modèles, les menus de navigations sont adaptés en fonction du type de l’utilisateurs connecté.

***COMMERCIAL***

***ADMIN***

***VENDEUR***

## Modèles de données

Les modèles de données ont été réalisés à l’aide du logiciel *Oracle SQL Developer Data modeler*.

### Explications

*Les explications suivantes se rapportent au modèle logique de données.*

Tout d’abord, nous trouvons les tables CANTONS et LOCALITES qui permettent de définir les provenances des personnes. La table CANTONS a été créée pour éviter la redondance d’informations au niveau de la table LOCALITES (ne pas devoir indiquer à chaque fois le nom et le code du canton dans celle-ci).

A la table LOCALITES sont liés 2 types de personnes : les clients (table CLIENTS) ainsi que les fournisseurs / représentants (table PERSONNES). Une association réflexive (table PERSONNES) permet de représenter le fait qu’un fournisseur peut avoir un représentant (son NUMERO serait alors renseigné dans le champ PERS\_NUMERO, qui n’est pas obligatoire).

La table PERSONNES est bien évidemment liée à la table VINS étant donné qu’un vin est directement lié à son fournisseur. Cette table contient non seulement des champs dédiés au code et nom du vin, mais aussi des champs permettant l’insertion d’une image pour chaque vin :

* **NOMING** 🡺 indique le nom de l’image (+extension de l’image)
* **TYPE\_MIME** 🡺 indique l’identifiant pour le format de données de l’image (*image/png* par exemple)
* **DATE\_MODIF\_IMG** 🡺 indique la date de la dernière modification de l’image
* **IMAGE** 🡺 contient l’image elle-même (fichier BLOB)

Etant donné que dans la réalité, plusieurs volumes de bouteilles sont possibles pour un même vin, nous avons créé un table VOLUMES qui recense l’ensemble des volumes disponibles pour les bouteilles de vin, en centilitres (50, 75, 150, etc…). Une bouteille de vin est donc une association entre un vin (table VINS) et un volume (table VOLUMES). Cela a donc donné lieu à la création d’une table associative BOUTEILLES qui reprend les NUMERO (clé primaires) des tables associées pour créer une bouteille, à laquelle s’ajoute un prix (colonne PRIX). Une contrainte d’unicité sur l’association des deux clés a été définie.

Un client (table CLIENTS) passe des commandes et est donc lié à la table COMMANDES. Chaque commande a un code ainsi qu’une date (à laquelle elle a été passée). Bien évidemment, il est possible que pour une même commande, plusieurs types de bouteilles de vin différents soient choisis. Cela signifie donc que la commande contiendra plusieurs lignes, une pour chaque type de bouteille. Cela a donc nécessité la création d’une seconde table associative, LIGNESCOMMANDES, liée à la table COMMANDES ainsi qu’à notre première table associative, BOUTEILLES. On a donc le détail de chaque ligne de la commandes (reprise des clés primaires pour la créer) ainsi qu’une colonne QUANTITE, qui permet au client de commander une bouteille en plusieurs quantités, comme dans un système traditionnel.

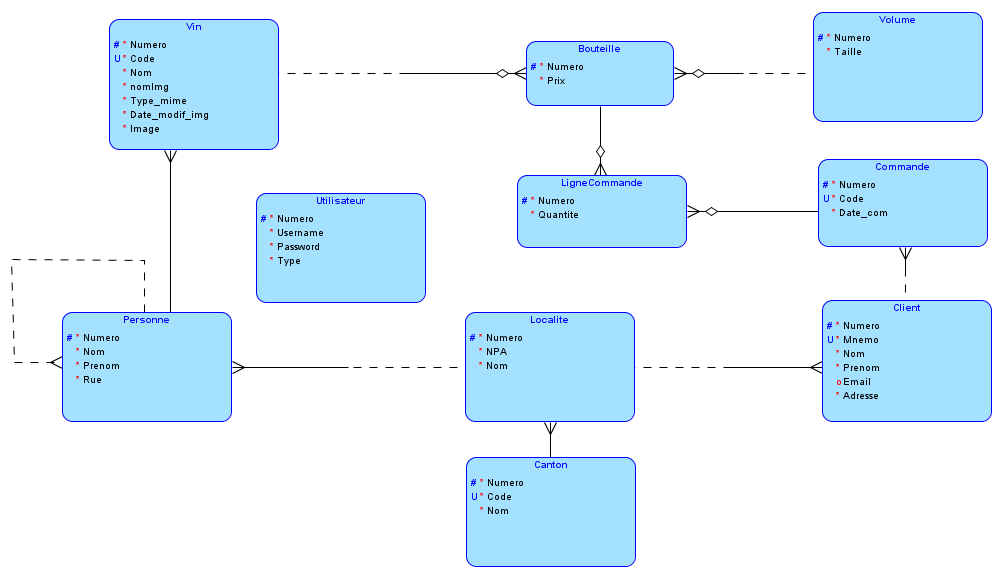
On peut voir que chaque table contient 4 même colonnes. Celles-ci sont créées pour des raisons de contrôle et de traçabilité. Voici leur signification :

* **CTRLAJUSER** 🡺 utilisateur ayant ajouté l’enregistrement dans la table
* **CTRLAJDATE** 🡺 date de l’ajout de l’enregistrement dans la table
* **CTRLMOUSER** 🡺 utilisateur ayant effectué la dernière modification sur l’enregistrement
* **CTRLMODATE** 🡺 date de la dernière modification sur l’enregistrement

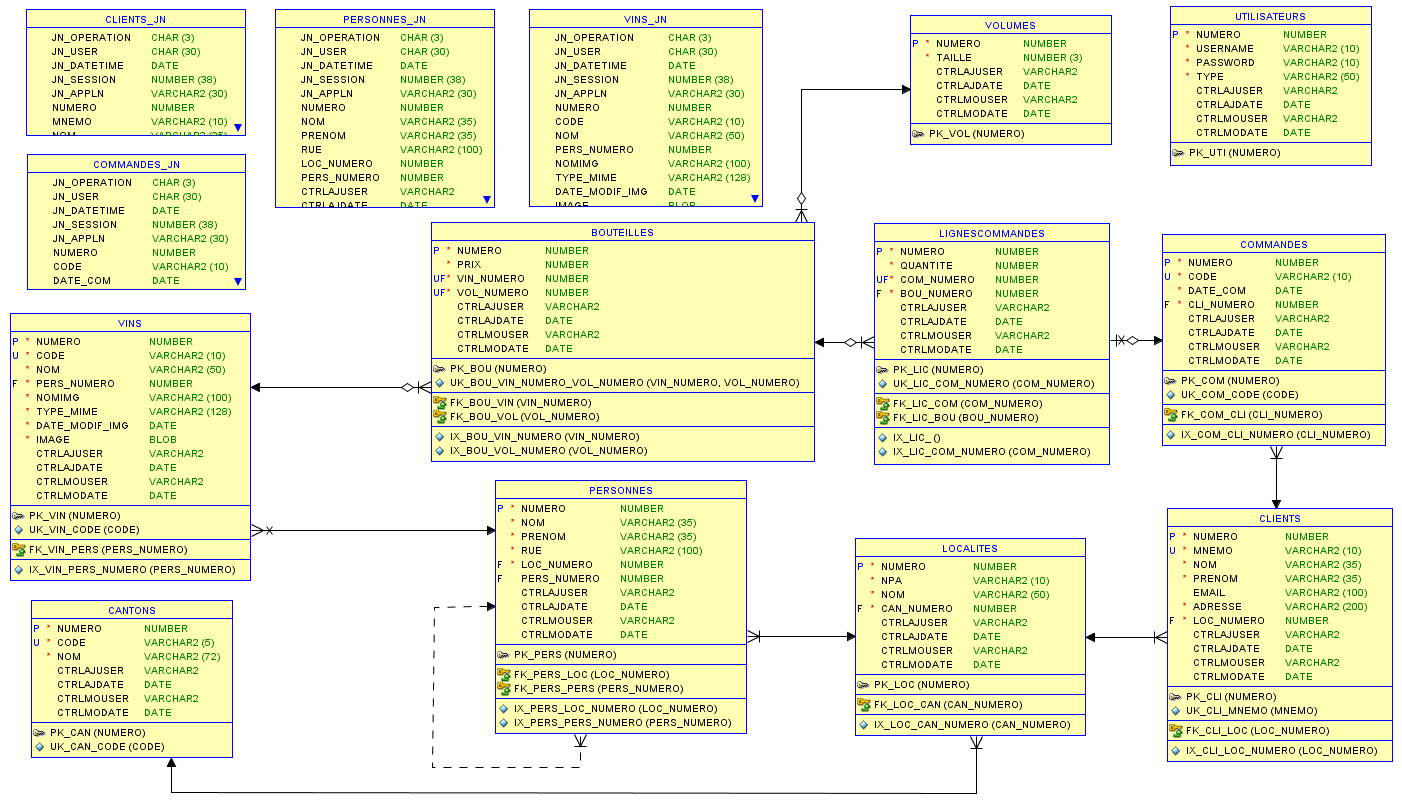
De plus, 4 tables de journalisation sont présentes, elles portent le suffixe \_JN après le nom de la table qu’elles journalisent. Une description des tables journalisées est donnée plus loin dans ce document.

Pour terminer, le modèle dispose d’une table UTILISATEURS faisant référence aux utilisateurs de l’applications (collaborateurs). Leur nom d’utilisateur, mot de passe (haché bien entendu) ainsi que le type de poste qu’ils occupent (droits et autorisations) doivent être renseignés.

### Modèle conceptuel de données / Logical Model



### Modèle logique de données / Relational Model



## Journalisation

Les informations des tables jugées importantes sont journalisées dans d’autres tables prévues à cet effet. Les tables journalisées sont les suivantes :

***- Clients - Personnes   
 - Commandes - Vins***

Cependant, les informations concernant les images liées aux vins ainsi que les images elles-mêmes ne sont pas journalisées car, en plus d’occuper beaucoup d’espace sur la base de données, nous avons jugé inutile de les consigner.

## Fonctionnalités choisies

Parmi les diverses fonctionnalités proposées dans le cas pratique, nous avons bien évidemment mis en place les fonctionnalités obligatoires :

* ***Etat classique et/ou état interactif***
* ***Formulaire simple et/ou formulaire tabulaire***
* ***Modèles d’authentification et d’autorisation***

De plus, nous avons intégré les fonctionnalités suivantes :

* ***Graphique***
* ***Image (logo)***
* ***Calendrier***

## Meilleure fonctionnalité

Plusieurs des fonctionnalités que nous avons mises en place pourraient être citées, comme les modèles d’autorisations en fonction du type d’utilisateur qui permettent de n’afficher que certains éléments du menu ou encore le tableau de bord s’affichant lorsqu’un utilisateur se connecte, lui permettant de visualiser des informations au sujet des ventes par exemple.

Cependant, la fonctionnalité que nous jugeons la meilleure est la génération automatique de codes pour l’ajout d’un vin. Nos modèles de données ne contiennent pas de table pour différencier les différents types de vins (blanc, rosé, rouge). Cependant, le code des vins en contient l’information. Il est construit de la manière suivante :

***BL*  
 GV *RG* XXXXX  
 *RS***

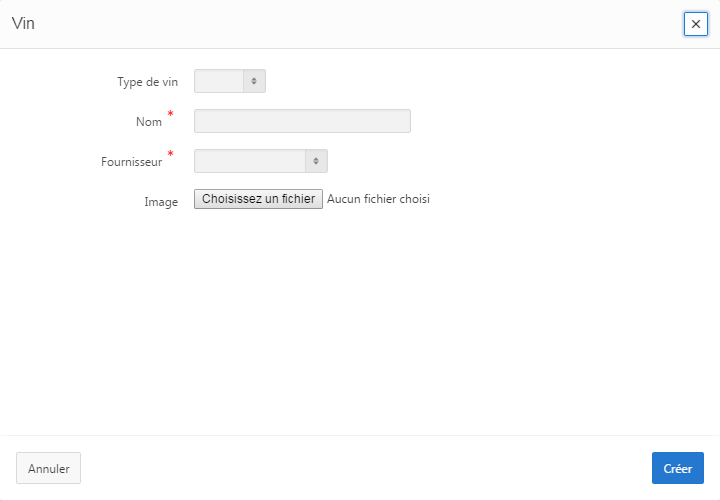
**Identique peu importe le type de vin. Cela représente une abréviation du nom de notre entreprise, *Gesto’Vin*.**

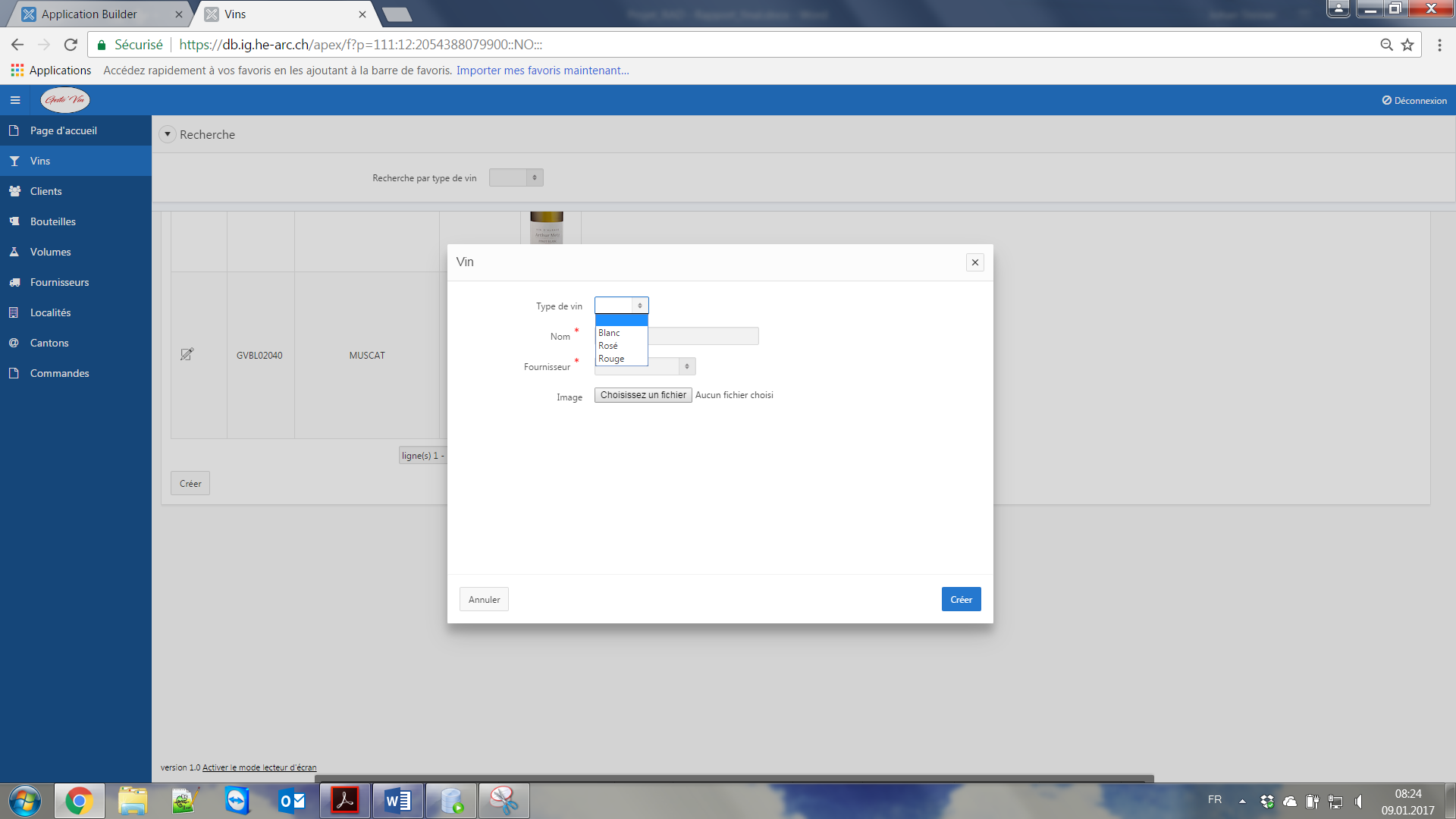
**En fonction du type de vin, les lettres sont différentes :   
BL = blanc / RG = rouge / RS = Rosé**

**Chaque vin a également un numéro à 5 chiffres qui suit les 4 premières lettres de son code.**

Il est donc important d’éviter les erreurs à ce niveau car une recherche a été mise en place en fonction du type de vin justement. Un code erroné ne permettrait donc pas de trouver le vin souhaité.

C’est pour cette raison que nous avons généré le code automatiquement. A l’insertion d’un vin, l’écran de l’utilisateur est le suivant :





L’utilisateur a simplement le choix du type de vin. En fonction de ce choix, l’un des trois codes à 2 chiffres est généré, devant lequel GV est placé.

Pour les 5 derniers chiffres, nous nous rendons dans notre base de données et recherchons le numéro à 5 chiffre le plus élevé parmi les vins du même type que celui choisi actuellement. A celui-ci est ajouté 10 afin de créer un nouveau numéro pas encore existant dans la base de données.

Cette génération est créée en utilisant une fonction PL/SQL.

## Problèmes rencontrés / Solutions trouvées

### Nommage de champ Date

Dans notre table *Commandes*, le champ censé contenir la date à laquelle a été passée la commande était nommé *Date*. En conséquence, nous avons rencontré des problèmes avec nos triggers lorsque ceux-ci souhaitaient créer un type de donné repris de ce champ (*.DATE%TYPE*).

Il a été nécessaire de modifier le nom du champ et générer à nouveau les triggers de cette table pour que le problème se résolve.

### Modification du modèle relationnel

Lorsque nous souhaitions mettre en place la fonctionnalité permettant de passer une commande, nous nous sommes aperçu d’une erreur présente dans notre modèle : une ligne de commande était reliée à la table *Vins* et non à la table *Bouteilles* (celle-ci est une table associative composé du vin et de la dimension de la bouteille choisie). Nous avons donc modifié notre modèle relationnel en conséquence mais le code SQL créé générait des erreurs lors de son exécution. Nous avons effectué plusieurs tentatives afin de recréer notre base de données correctement (régénération des modèles, désactivation de contraintes, etc…) mais les erreurs persistaient.

La solution a été d’exécuter les morceaux de codes les uns après les autres, dans un ordre différent et non en exécutant le script SQL comme généré par Oracle Data Modeler. Tout d’abord, nous avons créé les séquences ainsi que les tables, ensuite les triggers, les déclarations de nos packages puis leurs corps contenant les implémentations.

### Tablespace plein

Lors de modifications sur notre table *Vins*, nous avons eu un problème lié à un tablespace : celui-ci était plein et ne permettait plus de faire de modification dans la base de données. L’erreur nous indiquait qu’il était impossible d’étendre la capacité de stockage de celui-ci.

Pour pallier à ce problème, nous avons supprimé les champs *Noming*, *Type\_mime*, *Date\_modif\_img* et *Image* de notre table *Vins\_JN* car ceux-ci stockaient des informations relatives aux images insérées pour nous vins. Le champ *Image* stockait le fichier sous forme de Blob et occupait donc un espace considérable, en plus de ne pas être très utile en termes de traçabilité.

# Conclusion

L’application développée correspond à nos attentes. En effet, les fonctionnalités mises en place répondent aux besoins pour effectuer la gestion des vins, clients ou encore fournisseurs.

Durant le développement, de nombreux changements et imprévus ont cependant eu lieu, ce qui nous a poussé à reconsidérer certains de nos choix et à modifier des éléments définis auparavant suite à l’apparition de contraintes auxquelles nous n’avions pas pensé.

Notre méthode de travail, à savoir la création de modèles de données avant le développement de l’application, nous a permis de démarrer le projet avec une structure plus ou moins claire. Il a donc été plus simple de lier les éléments ensemble, créer les bons menus et les fonctions nécessaires.

Nous avons cherché à mettre l’utilisateur au centre, en développant une application ergonomique et dont l’utilisation est la plus simple possible pour que chacun puisse s’en servir, quel que soit son niveau de compétences. En page d’accueil par exemple, un tableau de bord permettant de visualiser quelques statistiques et un graphique (meilleur client, nombre de bouteilles vendues par moi, évolution du nombre de clients par mois, etc…) ou encore un calendrier a été créé. Cela évite par exemple d’arriver directement sur un tableau indiquant tous les produits disponibles à la vente, ce qui serait certainement moins plaisant.

Nous sommes cependant d’avis que l’application devra être améliorée au fil du temps, que de nouvelles fonctionnalités pourront être ajoutées, que les anciennes pourront être optimisées ou modifiées. Il sera nécessaire de constamment mettre à jour le système pour qu’il reste utilisable par *Gesto’Vin*. Cela représente un élément capital, car l’application a du potentiel à nos yeux.

# Références

Camus, Fabrice. *AGL : Données – MCD*, février 2016.

Camus, Fabrice. *AGL : Données – MLDR-MPD*, mars 2016.