Projet informatique intégré

# Présentation du domaine d’application

Dans l’idée de créer un bar à cafés, une borne sera utilisée pour permettre aux clients d’effectuer leurs commandes. Notre application aura comme fonctionnalités les suivantes :

* la commande de cafés (avec des ajouts possibles, nommés *toppings,* pour permettre un choix un peu plus personnel),
* un système de carte de fidélité reprenant certains avantages (permettant d’avoir un formulaire d’ajout/modification/suppression du client),
* un système de gestion des stocks pour le café, les toppings ainsi que les différents accompagnements,



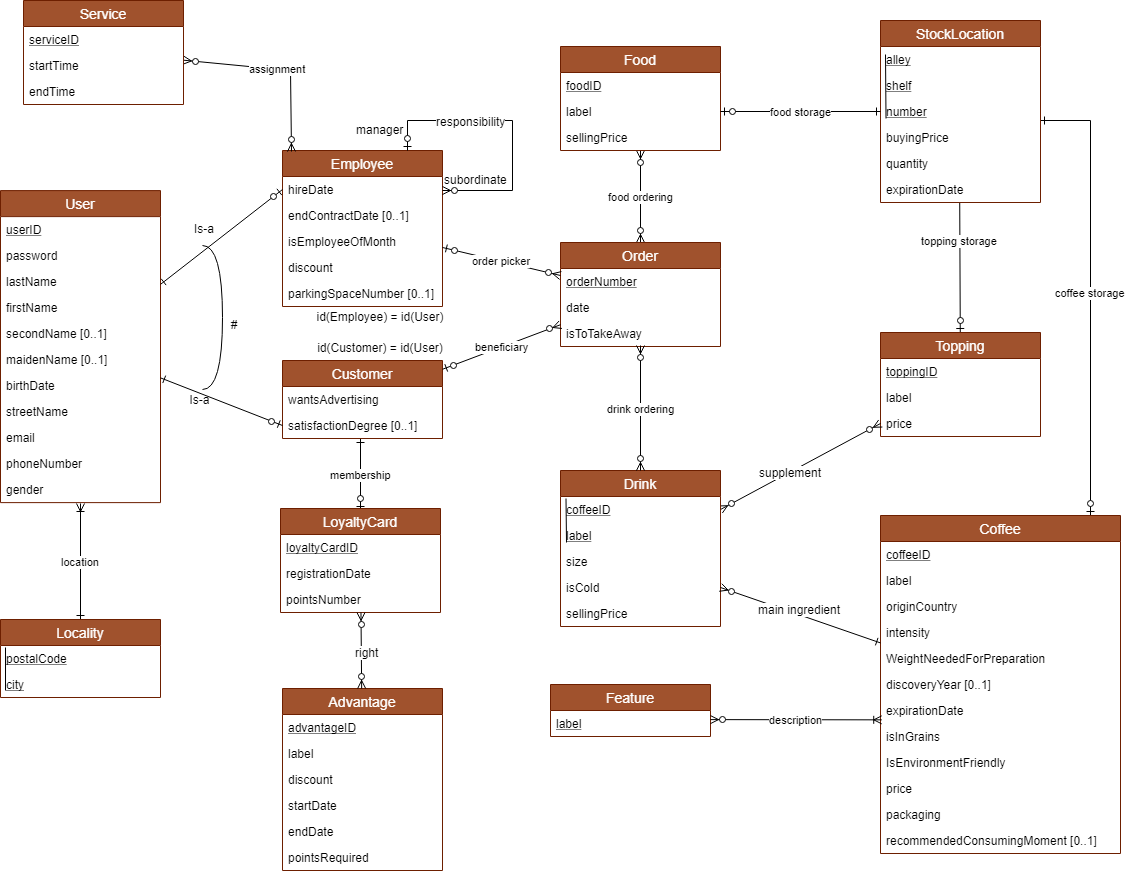
* une gestion des employés.

Les commandes sont sauvegardées dans la base de données ~~pour permettre aux clients de repasser une commande faite précédemment~~. Si le client décide de supprimer son compte, ses commandes sont tout de même gardées ~~pour pouvoir effectuer des statistiques chez le manager~~.

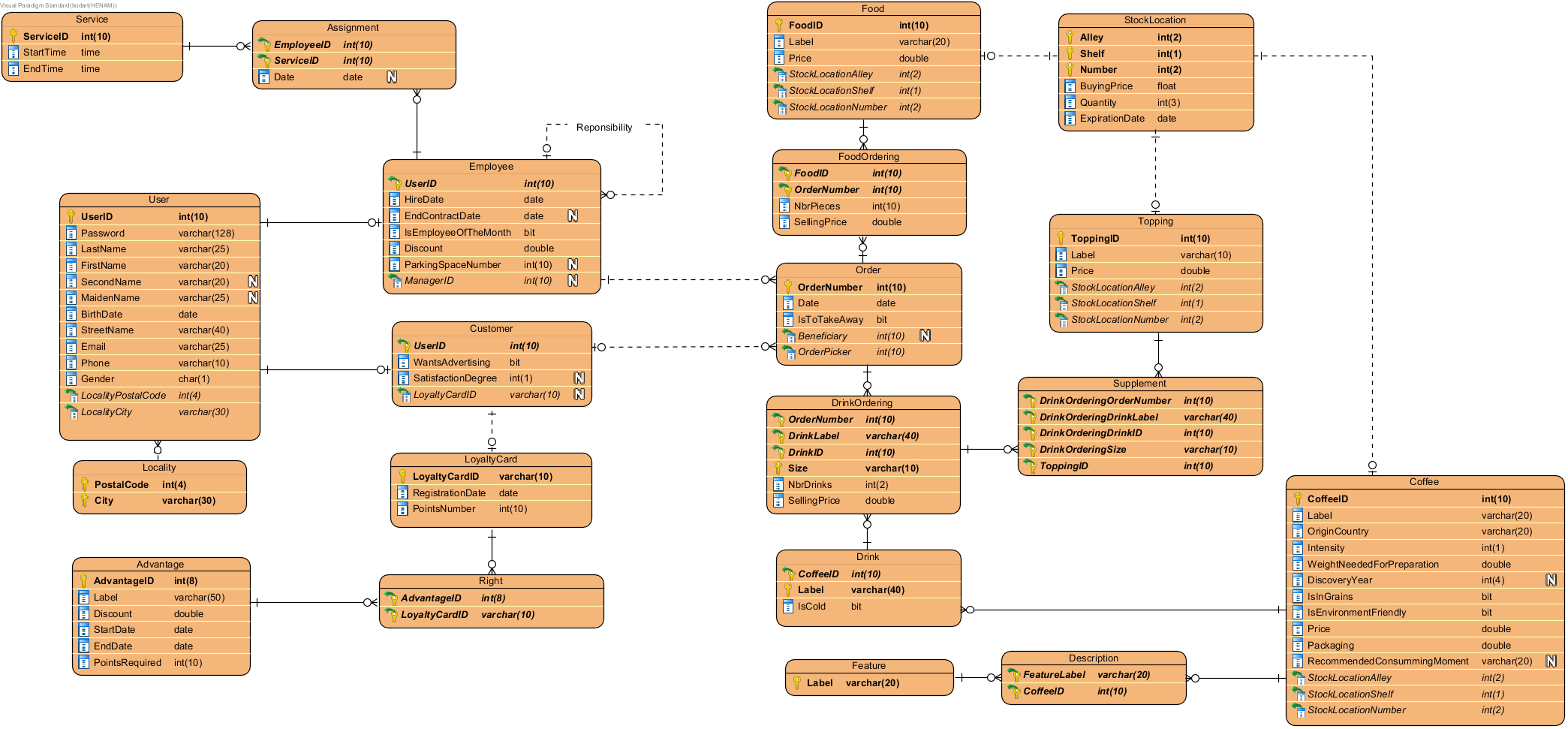
Lorsque le stock de certains produits est vide, ceux-ci ne sont plus disponibles comme choix de commande ; ce système permet de faire le calcul de prix en tenant compte des avantages de la carte de fidélité. Le système de gestion des stocks permet d’ajouter des nouveaux cafés, ~~d’en supprimer~~ et d’en mettre à jour.

~~Les employés peuvent gérer les stocks, les clients, ainsi que les horaires de leurs subalternes s’ils sont manager~~. Un formulaire de gestion des employés sera créé pour les managers afin d’effectuer les tâches de création, ~~de suppression et de modification~~ des employés.

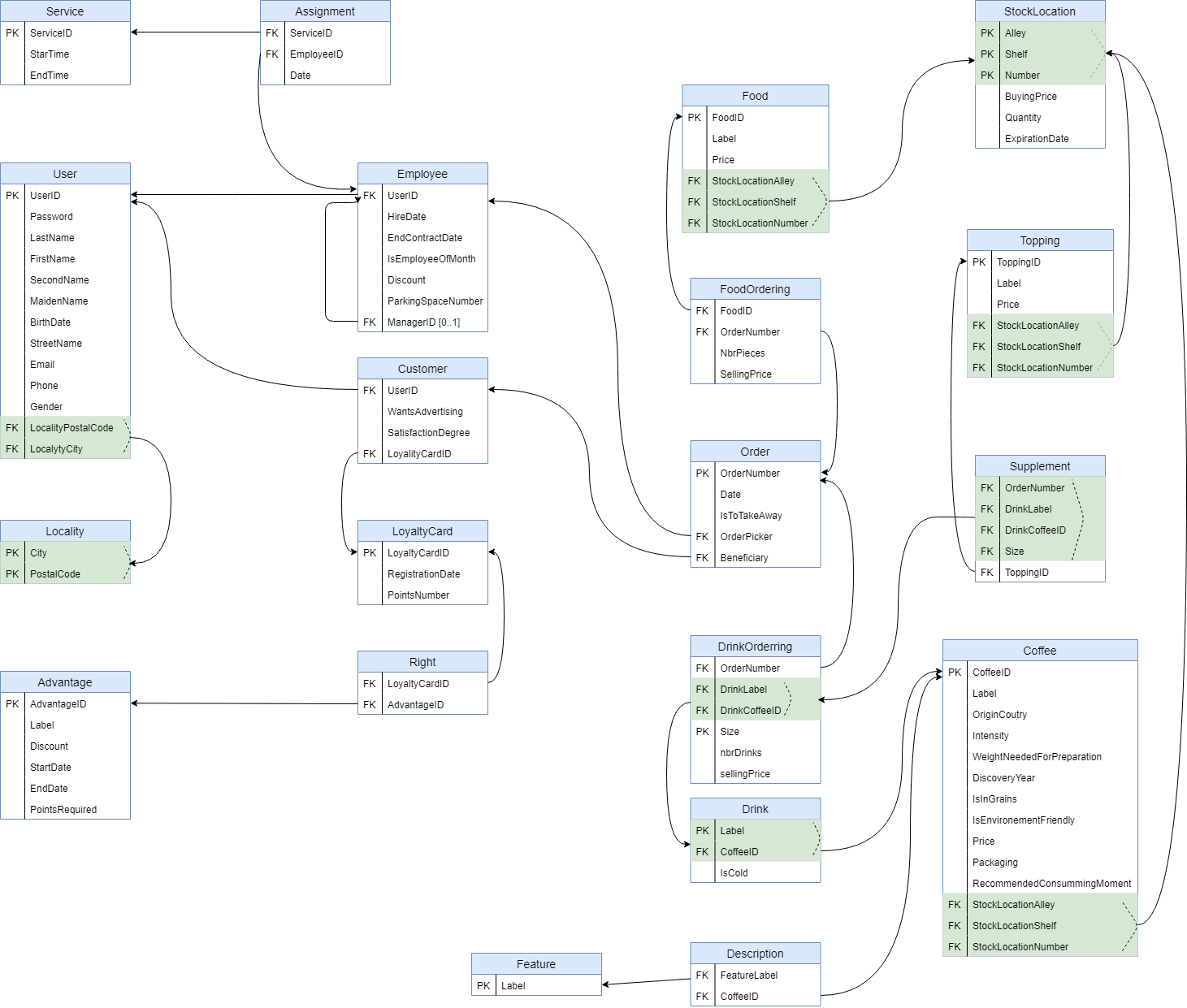
# Schéma entités-associations



# Schéma entités-relations



# Schéma logique des tables



# Spécification des tables

## User

Superclasse de client et employé.

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Type | Nullable ? | Description |
| UserID | Integer | / | Clé primaire de la table : un numéro unique. |
| Password | Varchar | / | Mot de passe du client. |
| LastName | Varchar | / | Nom de famille de la personne. |
| FirstName | Varchar | / | Prénom de la personne. |
| SecondName | Varchar | Oui | Deuxième prénom de la personne s’il en a un. |
| MaidenName | Varchar | Oui | Nom de jeune fille de l’utilisatrice si elle veut le renseigner. |
| BirthDate | Date | / | Date de naissance de la personne. |
| StreetName | Varchar | / | Rue et numéro de la personne. |
| Email | Varchar | / | Adresse mail unique de la personne. |
| PhoneNumber | Varchar | / | Numéro de téléphone unique de la personne. |
| Gender | Varchar | / | Genre de la personne. |
| LocalityPostalCode | Integer | / | Clé étrangère composée vers localité |
| LocalityCity | Varchar | / |

## Locality

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Type | Nullable ? | Description |
| PostalCode | Integer | / | Clé Code postal à 4 chiffres (on se limite aux communes namuroises). |
| City | Varchar | / | Nom de la ville de résidence. |

## Service

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Type | Nullable ? | Description |
| ServiceID | Integer | / | Clé primaire. Identifiant de la tranche horaire de travail. |
| StartTime | Time | / | Heure de début du service. |
| EndTime | Time | / | Heure de fin du service. |

## Assignment

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Type | Nullable ? | Description |
| EmployeeID | Integer | / | Clé étrangère. |
| ServiceID | Integer | / | Clé étrangère. |
| Date | Date | / | Date du service |

## Employee

La table employé permet d’avoir un formulaire. On pourra ajouter un nouvel employé dans la base de données lorsque celui-ci commence à travailler au Coff-IG. ~~Il sera possible de modifier son adresse, la date de fin de contrat s’il est prolongé par exemple. Lorsque le contrat est terminé, on supprime l’employé de la base de données. Tous les services qu’il a effectué seront également supprimés de la base de données après confirmation du manager.~~

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Type | Nullable ? | Description |
| UserID | Integer | / | Clé étrangère, obligatoirement égale à la clé primaire de User. Clé primaire de la table. |
| HireDate | Date | / | Date d’engagement de l’employé. |
| EndContractDate | Date | Oui | Date de résiliation de contrat (prévue pour les CDD, donc facultative). |
| IsEmployeeOfTheMonth | Bit | / | Booléen définissant l’employé comme employé du mois ~~(permet un éventuel pourcentage de réduction supplémentaire)~~. |
| Discount | Double | / | Pourcentage de réduction allouée à chaque employé sur une commande personnelle de café~~(peut varier en fonction de la responsabilité de l’employé)~~. |
| ParkingSpaceNumber | Integer | Oui | Numéro de parking éventuel réservé à l’employé (s’il vient en voiture). |
| ManagerID | Integer | Oui | Clé étrangère vers le manager dans le cas où l’employé est un subordonné. |

## LoyaltyCard

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Type | Nullable ? | Description |
| LoyaltyCardID | Varchar | / | Clé primaire. Numéro de téléphone du client identifiant la carte de fidélité. |
| RegistrationDate | Date | / | Date de délivrance de la carte au client. |
| PointsNumber | Integer | / | Nombre de points collectés sur la carte de fidélité (500 à la création). |

## Customer

La table client permet d’avoir un formulaire. On pourra ajouter un nouveau client et s’il veut une carte de fidélité, on pourra lui en créer une. On pourra modifier les informations du client telles que son adresse ou son numéro de téléphone. Si le client désire ne plus être enregistré, on pourra le supprimer de la base de données tout en gardant en mémoire les commandes qu’il a passées. Sa carte de fidélité sera également supprimée ainsi que les droits qui y étaient liés. Les avantages sont, eux, sauvegardés car ils sont liés à plusieurs cartes de fidélité. Le client pourra tout de même commander une fois la suppression effectuée.

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Type | Nullable ? | Description |
| UserID | Integer | / | Clé étrangère. Est obligatoirement égal à celui de User. |
| WantsAdvertising | Bit | / | Booléen représentant le choix de recevoir des publicités ou non. |
| SatisfactionDegree | Integer | Oui | Potentiel degré de satisfaction du client sur 5 (1/5 : pas satisfait ; 5/5 très satisfait). |
| LoyaltyCardID | Varchar | Oui | Clé étrangère. Numéro de carte de fidélité de la personne si elle décide d’en avoir une. Un client peut ne pas utiliser sa carte quand il effectue une commande. Un client peut aussi effectuer des commandes sans être renseigné comme client et sans avoir de carte. |

## Right

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Type | Nullable ? | Description |
| AdvantageID | Integer | / | Clé étrangère vers l’avantage. |
| LoyaltyCardID | Varchar | / | Clé étrangère vers la carte de fidélité. |

## Advantage

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Type | Nullable ? | Description |
| AdvantageID | Integer | / | Clé primaire. Identifiant de la réduction accordée. |
| Label | Varchar | / | Description de la réduction. |
| Discount | Double | / | X€ de réduction sur le montant total de la commande (lié aux nombre de points). |
| StartDate | Date | / | Date de début de l’action |
| EndDate | Date | / | Date de fin de l’action |
| PointsRequired | Integer | / | Nombre de points requis pour la réduction. |

## Order

On mémorise les commandes de tous les clients, enregistrés ou non.

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Type | Nullable ? | Description |
| OrderNumber | Integer | / | Clé primaire de la table. |
| Date | Date | / | Date à laquelle est passée la commande. |
| IsToTakeAway | Bit | / | Si la commande est à emporter ou non. |
| Beneficiary | Integer | Oui | Clé étrangère vers l’identifiant du client qui commande.  Peut être null si le client se supprime de la base de données. On garde cependant ses commandes ~~(pour calculer des statistiques. Ex : boisson la plus souvent commandée, …).~~ |
| OrderPicker | Integer | / | Clé étrangère vers l’identifiant de l’employé qui prépare la commande. ~~Si on supprime l’employé de la base de données, on garde les commandes qu’il a préparées.~~ |

## Drink

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Type | Nullable ? | Description |
| CoffeeID | Integer | / | Identifiant composé et clé étrangère vers le type de café utilisé dans la boisson. |
| Label | Varchar | / | Identifiant composé. |
| IsCold | Bit | / | Café froid ou chaud. |

## DrinkOrdering

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Type | Nullable ? | Description |
| OrderNumber | Integer | / | Identifiant composé et clé étrangère vers la commande à laquelle appartient la ligne de commande. |
| DrinkLabel | Varchar | / | Identifiant composé et clé étrangère vers la boisson. |
| DrinkID | Integer | / |
| Size | Varchar | / | Taille de la tasse de café (small, medium, large). |
| NbrDrinks | Integer | / | Nombre d’occurrences de la même boisson dans la ligne de commande. |
| SellingPrice | Double | / | Prix de vente de la boisson au moment de la commande, il peut être différent du prix du café et des toppings qui évoluent dans le temps. |

## Feature

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Type | Nullable ? | Description |
| Label | Varchar | / | Mot clé qui décrit le café pour le descriptif textuel. |

## Description

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Type | Nullable ? | Description |
| FeatureLabel | Varchar | / | Clé étrangère. |
| CoffeeID | Integer | / | Clé étrangère. |

## Coffee

La table café permet d’avoir un formulaire. On pourra ajouter une nouvelle variété de café dans la base de données pour la proposer à la vente. Le prix et le conditionnement pourront être modifiés (ainsi que les autres attributs). ~~Si on désire ne plus vendre un type de café, on pourra le supprimer.~~ Comme le prix est sauvegardé dans la commande, on pourra toujours garder une trace des ventes passées.

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Type | Nullable ? | Description |
| CoffeeID | Integer | / | Clé primaire de la table. |
| Label | Varchar | / | Nom du type de café utilisé dans la boisson. |
| OriginCountry | Varchar | / | Nom du pays d’origine pour le descriptif textuel. |
| Intensity | Integer | / | Intensité du café sur 5 (1 grain/5 : café doux ; 5 grains/5 : café fort). |
| WeightNeededForPreparation | Double | / | Nombre de grammes de grains de café pour préparer la boisson. Intervient dans la gestion du stock pour calculer la quantité stockée. |
| DiscoveryYear | Integer | Oui | Date de découverte pour le descriptif textuel. |
| IsInGrains | Bit | / | Pour savoir quel type de machine utiliser pour préparer la boisson. |
| IsEnvironmentFriendly | Bit | / | Café bio ou non. |
| Price | Double | / | Prix de vente du café (peut être différent d’une commande à la suivante). |
| Packaging | Double | / | Conditionnement des grains de café. |
| RecommendedConsummingMoment | Varchar | Oui | Moment de consommation préféré pour le descriptif textuel. |
| StockLocationAlley | Integer | / | Clé étrangère vers l’allée où est stocké le café. |
| StockLocationShelf | Integer | / | Clé étrangère vers l’étagère dans l’allée où est stocké le café. |
| StockLocationNumber | Integer | / | Clé étrangère vers le numéro sur l’étagère dans l’allée où est stocké le café. |

## Supplement

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Type | Nullable ? | Description |
| DrinkOrderingID | Integer | / | Clé étrangère vers l’identifiant de la ligne de commande de la boisson. |
| DrinkOrderingLabel | Varchar | / |
| DrinkOrderingOrderNumber | Integer | / |
| DrinkOrderingSize | Varchar | / |
| ToppingID | Integer | / | Clé étrangère vers l’identifiant du topping. |

## Topping

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Type | Nullable ? | Description |
| ToppingID | Integer | / | Clé primaire de la table. |
| Label | Varchar | / | Nom du topping (ex : lait, sucre, caramel, …). |
| Price | Double | / | Prix de vente du topping (peut être différent d’une commande à la suivante). |
| StockLocationAlley | Integer | / | Clé étrangère vers l’allée où est stocké le topping. |
| StockLocationShelf | Integer | / | Clé étrangère vers l’étagère dans l’allée où est stocké le topping. |
| StockLocationNumber | Integer | / | Clé étrangère vers le numéro sur l’étagère dans l’allée où est stocké le topping. |

## FoodOrdering

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Type | Nullable ? | Description |
| OrderNumber | Integer | / | Identifiant composé et clé étrangère vers la commande à laquelle appartient la ligne de commande. |
| FoodID | Integer | / | Identifiant composé et clé étrangère vers la nourriture. |
| NbrPieces | Integer | / | Nombre d’aliments identiques dans une ligne de commande. |
| SellingPrice | Double | / | Prix de vente de l’aliment au moment de la commande, il peut être différent du prix du café et des topping qui évoluent dans le temps. |

## Food

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Type | Nullable ? | Description |
| FoodID | Integer | / | Clé primaire de la table. |
| Label | Varchar | / | Nom de la nourriture (ex : cookies, tarte aux pommes, …). |
| Price | Double | / | Prix de vente de la nourriture (peut être différent d’une commande à la suivante). |
| StockLocationAlley | Integer | / | Clé étrangère vers l’allée où est stockée la nourriture. |
| StockLocationShelf | Integer | / | Clé étrangère vers l’étagère dans l’allée où est stockée la nourriture. |
| StockLocationNumber | Integer | / | Clé étrangère vers le numéro sur l’étagère dans l’allée où est stockée la nourriture. |

## StockLocation

~~Les quantités en stock sont calculables à partir des cardinalités qui relient Food, Topping et Coffee.~~ On pourra vérifier si les aliments sont toujours disponibles au moment de passer une commande si leur quantité n’est pas nulle. S’il n’y a plus de stock pour la commande, le champ sera grisé.

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Type | Nullable ? | Description |
| Alley | Integer | / | Clés primaires de la table. |
| Shelf | Integer | / |
| Number | Integer | / |
| BuyingPrice | Double | / | Prix d’achat de la marchandise au fournisseur. ~~La différence entre ce prix et le prix de vente permettra de calculer les bénéfices.~~ |
| Quantity | Integer | / | Quantité en stock d’une marchandise. |
| ExpirationDate | Date | / | Date de péremption du café, des toppings et de la nourriture. |

# Fonctionnalités de l’application

## Gestion des services des employés

~~Le manager pourra attribuer un service par jour à chaque employé. Ce service n’excéde pas les 8h.~~

## Gestion de fin de contrat

~~Le jour d’une fin de contrat, le manager reçoit une notification demandant si le contrat doit être prolongé ou s’il faut supprimer l’employé de la base de données. Cela sera géré avec un thread qui comparera quotidiennement la date du jour et la date de fin de contrat au lancement de l’application. Un autre thread permettra de mettre en attente la suppression jusqu’à la fin de journée si c’est l’option choisie par le manager dans la notification.~~

## Gestion de la carte de fidélité

Chaque client qui s’inscrit reçoit une carte de fidélité. Celle-ci lui donne accès à différents avantages en fonction des points qu’il cumule lors de ses commandes. Il reçoit 10 points par euro dépensé. Ces points sont sauvegardés sur sa carte de fidélité. Lorsqu’on est dans la période de validité d’un avantage et que la carte dispose de suffisamment de points, le client peut utiliser son avantage. Le nombre de point de l’avantage est déduit de sa carte.

Par exemple, si le client achète un café pour 5.25€, il reçoit 50 points. Ceux-ci se cumulent aux 270 points qu’il avait déjà sur sa carte. L’avantage « un café gratuit » nécessite 300 points et est valable du 1er avril au 15 avril. A la date du 5 avril, le client utilise son avantage et n’a plus que 20 points sur sa carte.

## Gestion de la satisfaction générale

~~Le manager pourra récupérer le degré de satisfaction de chaque client pour calculer la satisfaction moyenne de tous les clients ou afficher un graphique de répartition du nombre d’étoiles.~~

## Gestion du stock

Pour chaque emplacement de stock des toppings, cafés et nourritures, on retient la quantité disponible. On met à jour cette quantité après chaque commande, en fonction de la quantité consommée. Si la quantité est nulle, on rend indisponible l’aliment lors de la commande suivante. ~~Les employés sont prévenus lors d’une recherche et ils ont la possibilité de réapprovisionner le stock avec de nouvelles marchandises. La date d’expiration est contrôlée quotidiennement par un thread. Une notification est envoyée aux employés pour les avertir des articles périmés à retirer du stock.~~

## Gestion des prix

~~A chaque achat de marchandises auprès du fournisseur, le prix d’achat est mis à jour ainsi que le prix de vente de chaque topping, café et nourriture. Ce prix de vente peut évoluer dans le temps en fonction des réapprovisionnements. C’est pourquoi on retient le prix de vente réel dans la commande.~~

## Gestion des commandes

~~On retient toutes les commandes des clients pour calculer des bénéfices, ainsi que pour proposer au client des articles de ses commandes précédentes ou des articles populaires chez d’autres clients.~~

## Thread

Au lieu des threads définis au début du projet, nous avons choisi un autre thread plus visuel qui habille notre page d’accueil.

# Fonctionnalités de recherche

**Recherche numéro** 1 **(Aurélie)**

**Titre de la recherche** : Commandes passées par un client entre 2 dates

**Objectif de la recherche** : Afficher un récapitulatif des commandes passées par un client durant un certain laps de temps

**Jointure entre les tables suivantes** :

Table 1 : Customer

Table 2 : Order

Table 3 : DrinkOrdering

Table 4 : Drink

Table 5 : Food

Table 6 : FoodOrdering

**Entrées** :

|  |  |
| --- | --- |
| **Critère de recherche** | **Format** |
| Matricule du client | ~~JTextField (disable car récupéré au moment de la connexion)~~  JComboBox |
| Date de début | DatePicker |
| Date de fin | DatePicker |
| Emporté - sur place ~~- les deux~~ | JCheckbox |

**Sorties**(dans une JTable, en choisisant une commande, on remplace le contenu de la fenêtre avec le détail de la commande dans une JTable) :

|  |  |
| --- | --- |
| **Nom de la colonne** | **Nom de la table** |
| OrderNumber | Order |
| Date | Order |
| IsToTakeAway | Order |
| Price | DrinkOrdering + FoodOrdering |

**Sorties (dans une nouvelle JTable)** :

|  |  |
| --- | --- |
| Label | Drink |
| Est glacé | Drink |
| Taille | DrinkOrdering |
| Quantité | DrinkOrdering |
| Prix | DrinkOrdering |
| Label | Food |
| Quantité | FoodOrdering |
| Prix | FoodOrdering |

**Sorties (dans une nouvelle JTable) :**

|  |  |
| --- | --- |
| Identifiant | Topping |
| Label | Topping |
| Prix | Topping |

**Recherche numéro** 2 **(Maxime)**

**Titre de la recherche** : Avantage(s) du jour auquel(s) à droit un client

**Objectif de la recherche** : Listing des avantages valables ce jour-là pour un client et du nombre de points nécessaires pour en bénéficier

**Jointure entre les tables suivantes** :

Table 1 : Customer

Table 2 : LoyaltyCard

Table 3 : Right

Table 4 : Advantage

**Entrées** :

|  |  |
| --- | --- |
| **Critère de recherche** | **Format** |
| Matricule du client | ~~JTextField (disable car récupéré au moment de la connexion)~~  JComboBox |
| Date du jour | ~~JSpinner (disable car date système)~~  JTextField (disable car récupéré à la création de la fenêtre) |
| Pourcentage de réduction | JComboBox  ~~(10% - 20% - 50% - gratuit - tous pourcentages)~~  (tous les montants distints disponibles) |
| ~~Nombre de points~~  Tous – seulement accessibles – seulement inaccessibles | JRadioButton ~~(Uniquement avantages dont le nombre de points requis est atteint ou dépassé – Tous les avantages)~~  (En fonction du nombre de points et de la date de validité de l’avantage) |

**Sorties**:

|  |  |
| --- | --- |
| **Nom de la colonne** | **Nom de la table** |
| Label | Advantage |
| Remise | Advantage |
| StartDate | Advantage |
| EndDate | Advantage |
| PointsRequired | Advantage |
| ~~PointsNumber~~ | ~~LoyaltyCard~~ |
| ~~RemainingPoints~~ | ~~LoyaltyCard /PointsNumber -  Advantage /PointsRequired~~ |

**Recherche numéro** 3 **(Christophe)**

**Titre de la recherche** : Services futurs d’un employé

**Objectif de la recherche** : Permet l’affichage des heures de service d’un employé selon la durée choisie

**Jointure entre les tables suivantes** :

Table 1 : Employee

Table 2 : Assignment

Table 3 : Service

**Entrées** :

|  |  |
| --- | --- |
| **Critère de recherche** | **Format** |
| Liste des employés | JComboBox proposant le prénom + le nom |
| Date de début | ~~JSpinner~~  DatePicker |
| Durée | JComboBox (1 semaine – 15 jours – 1 mois) |

**Sorties**:

|  |  |
| --- | --- |
| **Nom de la colonne** | **Nom de la table** |
| Date | Assignment |
| StartTime | Service |
| EndTime | Service |

**Recherche numéro** 4

**~~Titre de la recherche~~** ~~: Gestion du stock selon la quantité restante et la date de péremption~~

**~~Objectif de la recherche~~** ~~: Vérifier les quantités en stock et leur date de péremption pour prévoir le réapprovisionnement~~

**~~Jointure entre les tables suivantes~~** ~~:~~

~~Table 1 : Coffee~~

~~Table 2 : StockLocation~~

~~Table 3 : Food~~

~~Table 4 : Topping~~

**~~Entrées~~**~~:~~

|  |  |
| --- | --- |
| **~~Critère de recherche~~** | **~~Format~~** |
| ~~Type de café~~ | ~~JList – sélection multiple  (latte – cappuccino – arabica - …)~~ |
| ~~Type de topping~~ | ~~JList – sélection multiple  (sucre – lait – caramel – chocolat – …)~~ |
| ~~Type de nourriture~~ | ~~JList – sélection multiple (cookies – tarte aux pommes - …)~~ |
| ~~Quantité minimal de café en stock~~ | ~~JTextField~~ |
| ~~Quantité minimal de topping en stock~~ | ~~JTextField~~ |
| ~~Quantité minimal de nourriture en stock~~ | ~~JTextField~~ |

**~~Sorties~~**~~:~~

|  |  |
| --- | --- |
| **~~Nom de la colonne~~** | **~~Nom de la table~~** |
| ~~Label~~ | ~~Coffee ou Food ou Topping~~ |
| ~~Quantity~~ | ~~StockLocation~~ |
| ~~ExpirationDate~~ | ~~StockLocation  (ligne en rouge quand article expiré –  ligne en orange quand article expire dans la semaine)~~ |