IUT Grand Ouest
Statistique et Informatique Décisionnelle
1re année de STID

SAE

Conception et implémentation d'une base de données

HOUZIER Baptiste
LADROU Victor
LESUEUR Romain
PONS-FOURNIER Amaury

Année universitaire : 2022 / 2023

Table des matières

1 Introduction

1.1 Présentation du projet

La petite commune nordique de Wulverdinghe, avec ses 322 habitants, est confrontée à un problème délicat. En raison de la coupure d'internet, le proxi local est désormais incapable d'utiliser les outils fournis par la centrale de Paris.

La solution proposée se décompose en quatre parties principales : la conception de base de données, le stockage des produits, la gestion des ventes et la génération de rapports. Chacune de ces parties vise à résoudre un aspect spécifique du problème et contribue à la mise en place d'un système efficace et fonctionnel.

Dans la première partie, nous proposons la mise en place d'une base de données utilisant un SGBD libre, qui répondra aux besoins du client. Cette base de données sera documentée et présentée au format UML.

La deuxième partie concerne le stockage des produits. Nous développerons un outil en ligne de commande permettant aux utilisateurs de gérer leur stock. Ils pourront ajouter, ranger et retirer des produits, ainsi que rechercher des produits spécifiques. De plus, un fichier de log sera généré pour enregistrer toutes les actions effectuées.

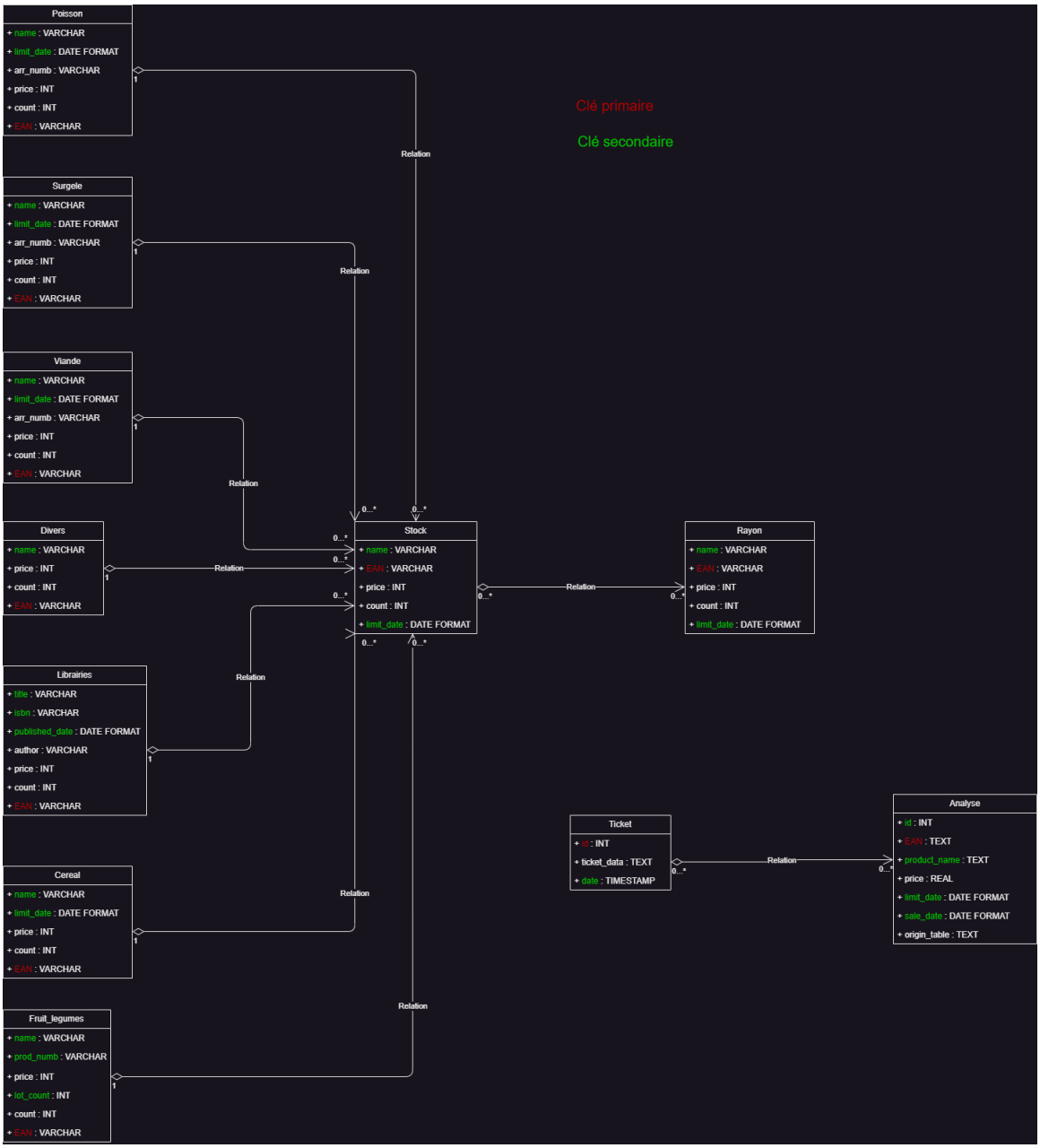
La troisième partie se concentre sur la vente. Nous compléterons l'interface graphique fournie, appelée Vente GUI, afin de permettre aux utilisateurs d'encaisser les clients. Un ticket de caisse au format txt sera généré et sauvegardé dans la base de données pour une analyse ultérieure.

Enfin, dans la quatrième partie, nous créerons un outil d'analyse des ventes effectuées, qui permettra de tirer des informations pertinentes à partir des données enregistrées.

Les données fournies comprennent des fichiers d'inventaire dans différents formats tels que JSON, CSV et TXT, couvrant différentes catégories de produits. Nous utiliserons ces données pour alimenter notre base de données et assurer une gestion efficace des stocks.

L'interface graphique Vente GUI, développée avec la librairie QT5 de Python, sera également utilisée pour faciliter les opérations de vente et l'émission des tickets de caisse.

1.2 UML


Poisson
+ name : VARCHAR
+ limit_date : DATE FORMAT
+ arr_numb : VARCHAR
+ price : INT
+ count : INT
+ EAN : VARCHAR
Clé primaire
Clé secondaire
Relation
Surgele
+ name : VARCHAR
+ limit_date : DATE FORMAT
+ arr_numb : VARCHAR
+ price : INT
+ count : INT
+ EAN : VARCHAR
Viande
+ name : VARCHAR
+ limit_date : DATE FORMAT
+ arr_numb : VARCHAR
+ price : INT
+ count : INT
+ EAN : VARCHAR
Divers
+ name : VARCHAR
+ price : INT
+ count : INT
+ EAN : VARCHAR
Stock
+ name : VARCHAR
+ EAN : VARCHAR
+ price : INT
+ count : INT
+ limit_date : DATE FORMAT
Rayon
+ name : VARCHAR
+ EAN : VARCHAR
+ price : INT
+ count : INT
+ limit_date : DATE FORMAT
Librairies
+ title : VARCHAR
+ isbn : VARCHAR
+ published_date : DATE FORMAT
+ author : VARCHAR
+ price : INT
+ count : INT
+ EAN : VARCHAR
Cereal
+ name : VARCHAR
+ limit_date : DATE FORMAT
+ price : INT
+ count : INT
+ EAN : VARCHAR
Fruit_legumes
+ name : VARCHAR
+ prod_numb : VARCHAR
+ price : INT
+ lot_count : INT
+ count : INT
+ EAN : VARCHAR
Ticket
+ id : INT
+ ticket_data : TEXT
+ date : TIMESTAMP
Analyse
+ id : INT
+ EAN : TEXT
+ product_name : TEXT
+ price : REAL
+ limit_date : DATE FORMAT
+ sale_date : DATE FORMAT
+ origin_table : TEXT

2 Problématique

Face à la coupure d'internet qui limite l'accès aux outils de la centrale de Paris, le magasin de proximité de Wulverdinghe est confronté à un défi majeur pour maintenir ses opérations. La problématique centrale de ce projet est donc la suivante : Comment mettre en place une solution locale efficace permettant au magasin de gérer son stock, d'encaisser les clients et d'analyser les ventes, en l'absence d'une connexion internet fiable ?

Cette problématique soulève plusieurs questions clés :

- Comment concevoir et mettre en place une base de données adaptée aux besoins du magasin, capable de gérer efficacement le stock des produits sans dépendre d'une connexion internet ?
- Comment développer un outil en ligne de commande permettant aux utilisateurs de gérer leur stock en effectuant des opérations telles que l'ajout, le rangement, le retrait et la recherche de produits ?
- Comment compléter l'interface graphique de vente existante afin de permettre aux utilisateurs d'encaisser les clients et de générer des tickets de caisse, tout en assurant la sauvegarde des données dans une base de données locale ?
- Comment créer un outil d'analyse des ventes qui permettra d'extraire des informations pertinentes à partir des données enregistrées, sans avoir recours à une connexion internet pour effectuer des requêtes externes ?
- La résolution de cette problématique nécessite une approche globale, en intégrant différentes parties du projet pour assurer la gestion efficace du stock, la vente des produits et l'analyse des ventes, tout en garantissant la fiabilité et la disponibilité des données sans une connexion internet.

3 Organisation du travail

Pour mener à bien ce projet, nous avons formé une équipe de 4 membres, répartis en fonction des diffénetes parties à réaliser. Voici l'organisation de l'équipe pour chaque partie du projet :

- ☐ Partie A (Conception de base de données) : Amaury et Victor étaient responsables de cette partie. Ils ont trravaillé conjointement pour proposer et mettre en place une base de données répondant aux besoins du client. Ils ont documenté cette base de données et l'ont présentée sous forme de diagramme UML dans le rapport.

- ☐ Partie B (Stockage) : Romain et Baptiste étaient chargés du développement de l'outil en ligne de commande permettant aux utilisateurs de gérer leur stock. Ils ont implémenté les fonctionnalités telles que l'ajout, le rangement, le retrait, la recherche de produits dans le stock et la possibilité de supprimer les produits périmés. De plus, ils ont généré un fichier de log pour enregistrer toutes les actions effectuées.

- ☐ Partie C (Vente) : Romain et Baptiste ont également pris en charge cette partie. Ils ont complété l'interface graphique fournie, appelée Vente GUI, afin de permettre aux utilisateurs d'encaisser les clients. Ils ont mis en place la génération d'un ticket de caisse au format txt et ont assuré que ces tickets de caisse étaient sauvegardés dans une base de données pour une analyse ultérieure. De plus, lorsqu'un produit est vendu, la base de données est mise à jour pour refléter la nouvelle quantité de stock disponible.

- ☐ Partie D (Compte Rendu) : Romain était responsable de cette partie. Il a développé un outil permettant d'analyser les ventes effectuées et d'en tirer des informations pertinentes. Cet outil fournit des statistiques et des rapports basés sur les données des produits vendues (Aléatoire).

Cette répartition des tâches a permis à chaque membre de l'équipe de se concentrer sur une partie spécifique du projet, favorisant ainsi une meilleure productivité et une expertise spécialisée. Une communication régulière a été maintenue entre les membres de l'équipe pour garantir une cohérence et une intégration harmonieuse des différentes parties du projet.

4 Explication des différentes parties du projet

Dans ce chapitre nous vous expliquerons comment et à quoi sert ce que nous avons fait.

4.1 Partie A : conception de base de données

Ce code effectue plusieurs opérations liées à la gestion d'un inventaire. Voici une explication détaillée de chaque partie du code :

Nous avons d'abord importer plusieurs bibliothèques :

- datetime (est utilisée pour manipuler et effectuer des opérations sur les dates et les heures)
- sqlite3 (est utilisée pour interagir avec des bases de données SQLite en Python. Elle permet d'établir des connexions, d'exécuter des requêtes SQL, de gérer les transactions et de récupérer les résultats des requêtes)
- package as pk (permet d'importer des fonctions pour aller plus vite)
- pandas as pd (utilisée pour l'analyse et la manipulation de données)
- csv (facilite la manipulation des fichiers CSV pour lire, écrire et personnaliser les opérations sur ces types de fichiers)
- random (permet de travailler avec des nombres aléatoires et effectuer des opérations liées au hasard)

Ces bibliothèques sont nécessaires pour les opérations de traitement des données, la gestion de bases de données SQLite, la manipulation de fichiers CSV et JSON, ainsi que la génération de nombres aléatoires.

Par la suite nous avons commencé à charger les fichiers contenant les données de l'inventaire. Les fichiers de données, tels que "poisson.csv", "surgele.csv", "viandes.csv", "cereal.json", "librairie.json", "divers.txt" et "fruitslegumes.txt", sont chargés dans des variables correspondantes. Ces fichiers contiennent les informations sur les produits de différents types.

Nous avons ensuite supprimer les premières lignes (en-têtes) de chaque fichier CSV chargé précédemment pour faciliter le traitement des données.

Nous avons également convertit la colonne "authors" en chaîne de caractères. La colonne "authors" du fichier JSON "librairie.json" est convertie en une chaîne de caractères. Cela est nécessaire car la colonne "authors" peut contenir une liste d'auteurs, et la conversion permet de la représenter sous forme de chaîne de caractères. Nous avons créer différentes tables sont créées dans la base de données pour stocker les informations sur les produits de différents types tels que le poisson, les surgelés, les viandes, les céréales, les livres, les articles divers, les fruits et légumes, etc. Les colonnes de chaque table sont spécifiées avec leurs types de données respectifs.

Nous avons du insérer les données dans les tables respectives et insérer l'inventaire dans le stock. Les données extraites des fichiers ont été insérée dans les tables correspondantes de la base de données. Nous avons utilisé des requêtes SQL d'insertion pour ajouter les données dans les tables. Les données des différentes tables sont transitées vers la table "stock". Cela permet de regrouper toutes les informations sur les produits dans une seule table pour faciliter la gestion de l'inventaire.

4.2 Partie B : Stockage

Dans cette partie, nous avons plusieurs fonctions pour effectuer différentes actions de gestion de stock.

La fonction "logaction()" permet d'enregistrer les actions effectuées dans un fichier de journal ("stock.log"). Chaque fois qu'une action est effectuée, elle est enregistrée avec la date et l'heure.

La fonction "rentrerproduit()" permet d'ajouter un nouveau produit au stock ou de mettre à jour la quantité d'un produit existant. L'utilisateur est invité à fournir les informations sur le produit telles que le nom, l'EAN (code-barres), la quantité, le prix unitaire, et éventuellement la date limite de consommation. Si le produit existe déjà dans la base de données, sa quantité est mise à jour. Sinon, un nouveau produit est ajouté.

La fonction "rangerproduitstock()" permet de déplacer un produit du rayon vers le stock. L'utilisateur est invité à entrer le code EAN du produit à déplacer. Si le produit existe dans le rayon et qu'il y a une quantité disponible, l'utilisateur est invité à entrer la quantité à transférer. La quantité est mise à jour dans le rayon et dans le stock. Si le produit n'existe pas dans le rayon, un message d'erreur est affiché.

La fonction "rangerproduitrayon()" permet de déplacer un produit du stock vers le rayon. L'utilisateur est invité à entrer le code EAN du produit à déplacer. Si le produit existe dans le stock et qu'il y a une quantité disponible, l'utilisateur est invité à entrer la quantité à transférer. La quantité est mise à jour dans le stock et dans le rayon. Si le produit n'existe pas dans le stock, un message d'erreur est affiché.

La fonction "retirerproduit()" permet de supprimer un produit du stock. L'utilisateur est invité à entrer le code EAN du produit à supprimer. Si le produit existe dans le stock, il est supprimé de la base de données.

La fonction "rechercherproduit()" permet de rechercher un produit dans le stock. L'utilisateur est invité à entrer le code EAN du produit à rechercher. Si le produit est trouvé dans le stock, ses informations sont affichées. Sinon, le produit est recherché dans le rayon. Si le produit est trouvé dans le rayon, ses informations sont affichées. Sinon, un message indiquant que le produit n'est pas présent est affiché.

La fonction "logactionpertes()" permet d'enregistrer les détails des produits périmés dans un fichier de journal des pertes ("pertes.txt"). Les produits périmés sont récupérés à la fois dans le rayon et dans le stock. Les détails des produits supprimés sont enregistrés dans le fichier de journal.

Enfin, la fonction "supprimerproduitsperimes()" permet de supprimer les produits périmés à la fois du rayon et du stock. Les produits périmés sont récupérés à partir de la date actuelle et supprimés de la base de données. Les détails des produits supprimés sont enregistrés dans le fichier de journal des pertes.

La fonction principale "main()" affiche un menu d'options pour l'utilisateur. En fonction du choix de l'utilisateur, la fonction correspondante est appelée pour effectuer l'action demandée.

4.3 Partie C : Vente

La partie C est consacrée à la vente. Le but de celle-ci est de compléter une interface graphique qui nous a été fournie se nommant « Vente GUI ». Cette interface graphique nous servira pour encaisser les commandes des clients. À la fin, nous devons arriver à générer un ticket au format TXT que nous sauvegarderons dans une base de données afin que nous puissions faire des analyses.

Nous avions fait notre partie C dans un script séparé intitulé « venteGUI ». Tout d'abord, nous importons les librairies nécessaires, en particulier celle se nommant « QT5 ». Cette librairie sert à créer des interfaces graphiques.

L'interface graphique que nous devions créer est composée de plusieurs parties. Une zone de saisie, du texte d'information, du texte représentant le ticket en cours, un bouton de validation d'article et un bouton de validation de ticket.

La fonction « validation-article » permet de, comme son nom l'indique, de valider un article que l'utilisateur a entré. Elle va d'abord vérifier si l'article saisi existe en allant le chercher dans le rayon grâce à son EAN. Si l'article est présent, alors les informations de celui-ci (article, nom du produit, prix et EAN) seront ajoutées au ticket de caisse. Sinon, alors nous recevons un message disant comme quoi le produit est introuvable. Ce processus est répétable jusqu'à épuisement des stocks. Après les commandes terminées, toutes les informations concernant les produits achetés par l'utilisateur sont enregistrées dans une table « analyse ».

Ensuite, nous avons la fonction « update-stock-after-sale ». Elle tout simplement mettre à jour les stocks seulement quand l'utilisateur aura fini sa commande et qu'il aura cliqué sur END. La fonction va nous donner la quantité restante des produits présents dans les rayons. Quand l'utilisateur commande un produit alors sa quantité sera réduit de 1 dans la table rayon. Mais si le produit est en rupture dans le rayon, alors la fonction vérifiera si ce produit est présent en stock. Deux possibilités sont possibles, si le produit est en stock, alors il sera transféré du stock au rayon. À l'inverse, le produit sera totalement supprimé du rayon.

La fonction « extract-ean-list » va juste nous permettre de récupérer tous les EAN qui sont présents dans les tickets de commandes des clients. Ils seront stockés dans une liste.

Finalement la fonction « validation-ticket », cette fonction fonctionnera quand l'utilisateur aura fini sa commande en cliquant sur END. Elle va enregistrer toutes les informations de la commande ainsi que la date et heure précise à laquelle l'utilisateur aura terminé sa commande. Ce ticket va être enregistré dans la base. Et pour finir, les stocks seront mis à jour.

4.4 Partie D : Compte Rendu

Nous avons fait notre partie D en 2 étapes. La première en python où nous avons généré un fichier CSV sur des ventes de produits aléatoires. Ensuite, nous sommes allés sur R Studio pour faire une analyse du fichier créé.

4.4.1 Partie python

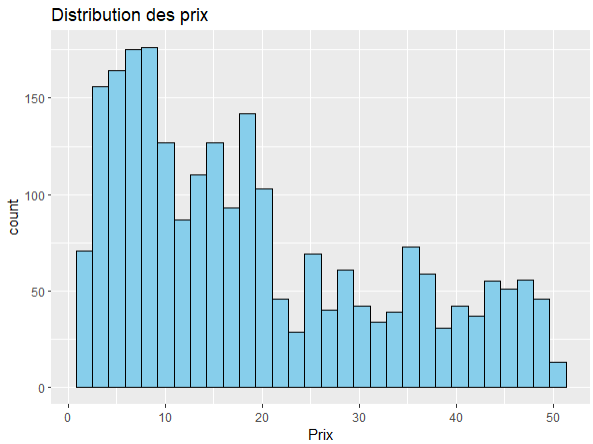
Dans cette partie, nous allons voir comment nous avons créé notre fichier de sortie sur des ventes. Pur cela, nous avons pris aléatoirement un total de 980 / 2355 produits que nous avons insérer dans une table « analyse ». Pour pouvoir faire une meilleure analyse sur ces produits, nous avons généré une date de ventes pour chaque produit de la table. Le fichier contient alors toutes les informations que nous souhaitons : EAN, nom du produit, son prix, sa date limite de consommation, sa date de vente et le type du produit

4.4.2 Partie R

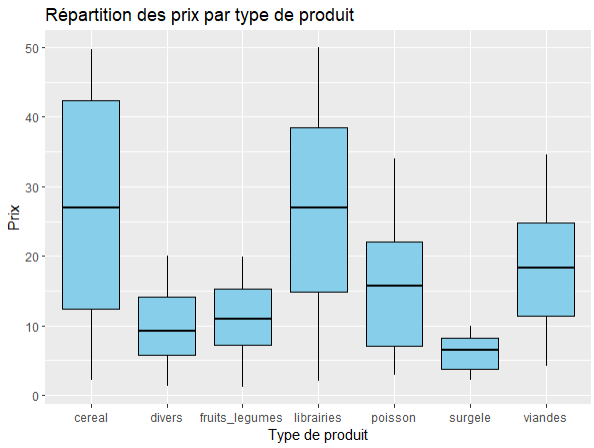
Après avoir créé le fichier pour l'analyse des ventes, nous pouvons enfin en faire son analyse. Pour cela, nous sommes partis sur R Studio pour pouvoir faire les graphiques nécessaires. Avant de réaliser les graphiques, nous avons quelques étapes à faire. Commençons par l'importation des librairies utiles pour les graphiques. Ensuite, nous importons les données du fichier CSV dans un dataset qu'on nommera « data ». Nous renommons également au passage les colonnes. Enfin, nous avons appliqué le bon format pour les dates afin de ne pas avoir de problème lors de l'analyse, et nous avons créé une variable qui indique le nombre de jours restant avant qu'un produit ne soit périmé.

Nous allons maintenant analyser rapidement les differents graphiques sur les ventes du commerce.

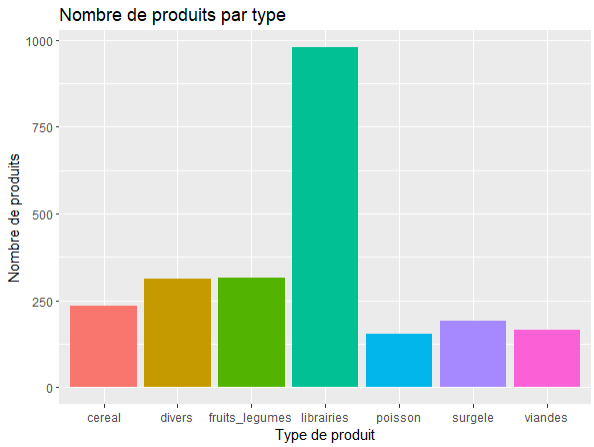
1. Le premier graphique représente la distribution des prix. Nous pouvons constater que la majorité des produits ont un prix inférieur à 20 €. Et nous remarquons qu'il y a de moins en moins de produits quand le prix augmente.



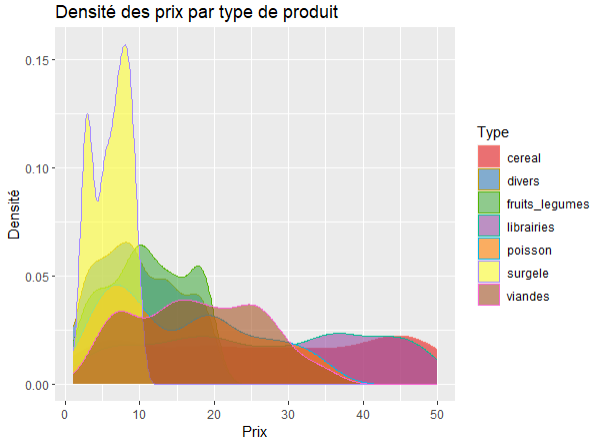
2. Le deuxième graphique représente la répartition des prix par type de produit. Nous voyons que la répartition des prix pour la catégorie « cereal » et « librairies » est très élevé. Un peu moins pour les catégories « poisson » et « viandes ». Et nous avons les produits des catégories « divers », « fruits-legumes » et « surgele » dont les prix sont les plus rapprochés.



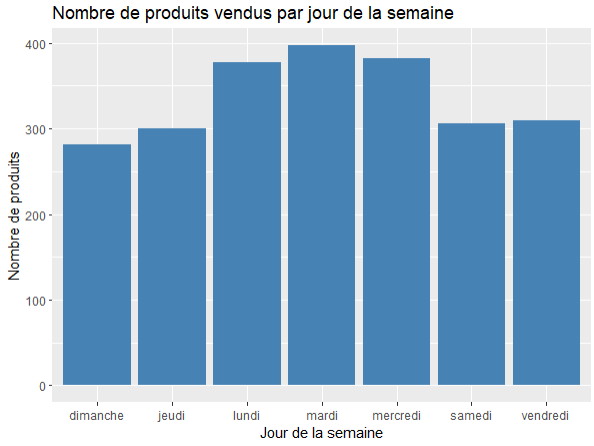
3. Le troisième graphique représente le nombre de produits par type. Nous constatons qu'il y a le plus de produit dans la catégorie « librairies ». Les autres catégories ont plus ou moins le même nombre de produits chacun. En revanche, la catégorie « poisson » est celle qui comporte le moins de produit.



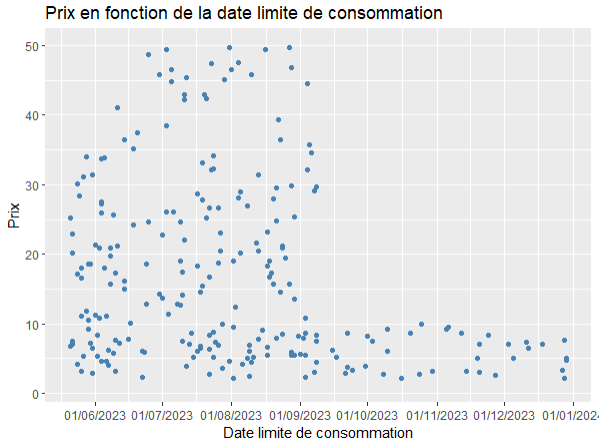
4. Ce graphique montre la densité des prix par type de produit. La catégorie de produit ayant la plus forte densité de prix est la catégorie « surgele ». Ensuite les produits des catégories « divers », « fruit-legumes » ont une densité assez similaire. De même pour les catégories « poisson » et « viande ». Enfin les catégories « divers » et « cereal » sont celles dont les prix ont la moins forte densité.



5. Le graphique suivant représente le nombre de produits vendus par jour de la semaine. Il y a trois jours où les produits ont été le plus vendu, c'est le lundi, mardi et mercredi. Parmi ces trois-là, c'est le mardi qu'il y a eu le plus de ventes. A l'inverse, c'est le dimanche où les clients achètent le moins.



6. Finalement, le dernier graphique représente le prix en fonction de la date limite de consommation. Nous remarquons que les produit ayant une date limite de consommation proche, ont un prix plus élève que les autres produits.



5 Résolution

La problématique de ce projet était la suivante :
Comment mettre en place une solution locale efficace permettant au magasin de gérer son stock, d'encaisser les clients et d'analyser les ventes, en l'absence d'une connexion internet fiable ?

Cette problématique amena plusieurs questions et nous les avions toutes répondus avec succès.

Nous avons commencé par mettre en place une base de données adaptée aux besoins du magasin, qui est capable de gérer efficacement le stock des produits.

Ensuite, nous avons réussi à développer un outil en ligne de commande permettant aux utilisateurs de gérer leur stock en effectuant des opérations telles que l'ajout, le rangement, le retrait et la recherche de produits du magasin.

Après avoir complété l'interface graphique de vente existante, permettant aux utilisateurs d'encaisser les commandes des clients et de générer des tickets de caisse, tout en sauvegardant des données des tickets dans une base de données locale.

Enfin, nous avons créé un outil d'analyse des ventes qui permettait d'extraire des informations pertinentes à partir des données enregistrées a des fins analytiques.

6 Difficulté

Le projet en lui-même est simple, il suffisait de faire de la gestion d'une base de données. Donc la créer et faire des modifications à l'intérieur tel que l'ajout ou la suppression d'éléments.

Le seul point négatif, ce n'est pas vraiment une difficulté, mais c'est que le projet est assez long à terminer.

7 Conclusion

En conclusion, ce projet nous a permis de développer une solution locale efficace pour la gestion des stocks, l'encaissement des clients et l'analyse des ventes dans la petite commune nordique de Wulverdinghe. Face à la coupure d'Internet qui limitait l'accès aux outils de la centrale de Paris, nous avons réussi à mettre en place un système fonctionnel en utilisant plusieurs parties principales.

La première partie du projet concernait la conception d'une base de données adaptée aux besoins du magasin. Nous avons utilisé un SGBD libre et documenté la base de données au format UML pour assurer sa compréhension et sa maintenance.

La deuxième partie était dédiée au stockage des produits. Nous avons développé un outil en ligne de commande permettant aux utilisateurs de gérer leur stock en ajoutant, rangeant et retirant des produits, ainsi qu'en effectuant des recherches. Un fichier de log a été généré pour enregistrer toutes les actions effectuées.

La troisième partie était axée sur la vente. Nous avons complété l'interface graphique existante, appelée Vente GUI, pour permettre aux utilisateurs d'encaisser les clients. Nous avons également généré des tickets de caisse au format TXT et les avons sauvegardés dans la base de données pour une analyse ultérieure. La quantité de stock disponible a été mise à jour lors de chaque vente.

Enfin, dans la quatrième partie, nous avons créé un outil d'analyse des ventes qui nous a permis d'extraire des informations pertinentes à partir des données enregistrées. Cet outil a fourni des statistiques et des rapports basés sur les produits vendus.