Name ☺ : Sundus AL KEBSI
Date 18: Dimanche 18 avril 2021

Interaction Homme-Machine L3S5 “La Banque”

Abstrait : Le projet porte sur une implémentation d’un marché pour un secteur bancaire «La Banque». C'est une application utilisée par les banquiers, plus précisément par les directeurs de la banque pour voir l'évaluation de leurs ventes. Où tout est organisé à la demande du client, chaque agence a sa propre évaluation, à l'intérieur de cette évaluation, tout canal a ses propres familles et chaque famille a ses propres versions de produits. tandis que pour le directeur général, il y a une vue générale de toutes les agences.

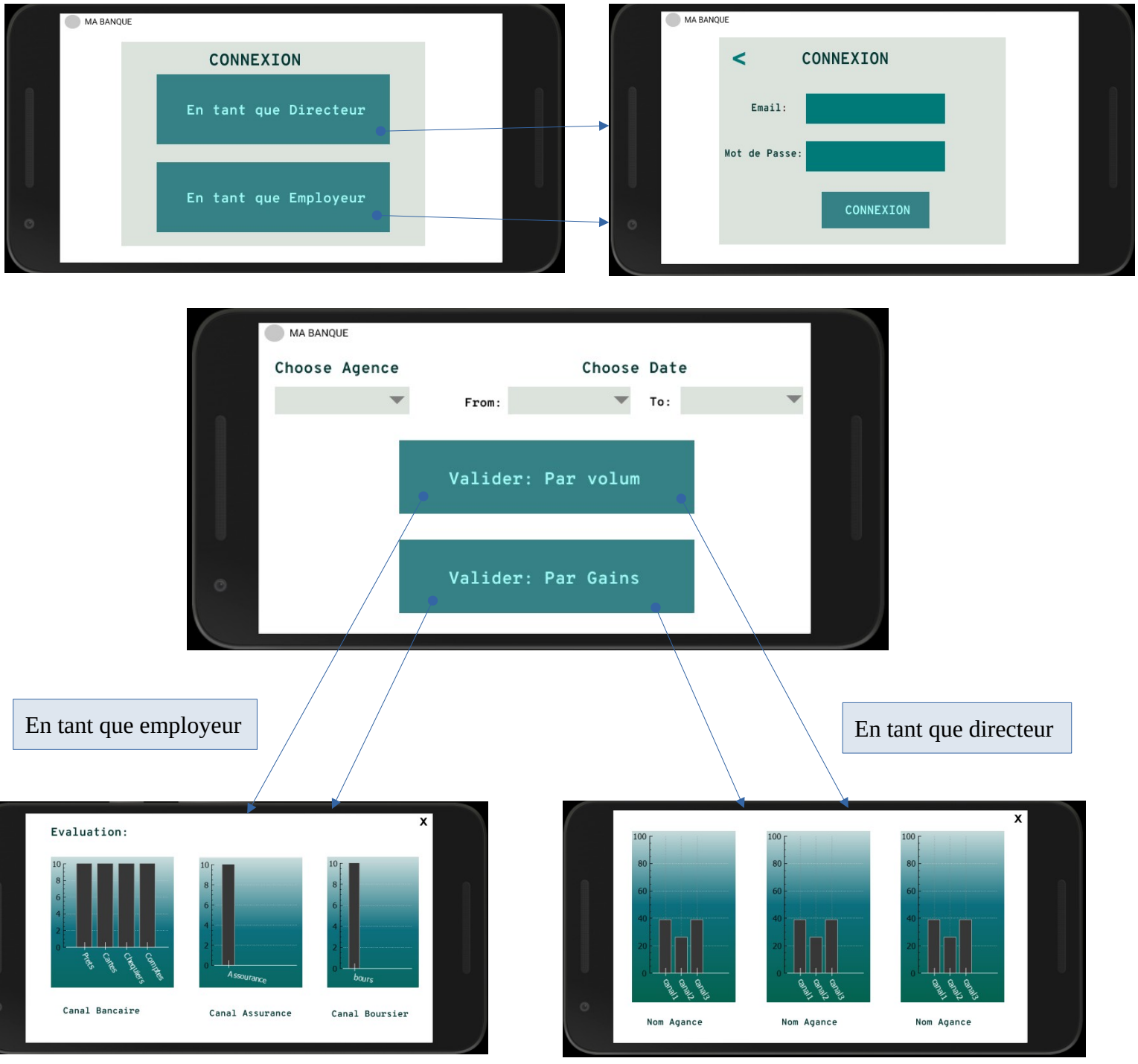
Mode d'utilisation :

Pour faire fonctionner ce programme, un chemin absolu du dossier "database" doit être ajouté à la classe mainWindow dans le fichier mainWindow.h, aux lignes 35 et 36, dans les variables déclarées pathabs et pathChar.

```
35      string pathabs = "C:/Users/sundu/Documents/banque/database/";
36      const char *pathChar = "C:/Users/sundu/Documents/banque/database/agence1/";
```

Contents :

1 Wireframe


MA BANQUE
CONNEXION
En tant que Directeur
En tant que Employeur
MA BANQUE
<
CONNEXION
Email:
Mot de Passe:
CONNEXION
MA BANQUE
Choose Agence
Choose Date
From:
To:
Valider: Par volum
Valider: Par Gains
En tant que employeur
En tant que directeur
Evaluation:
Prets
Cartes
Chequiers
Comptes
Assourance
bours
Canal Bancaire
Canal Assurance
Canal Boursier
canal1
canal2
canal3
Nom Agance
Nom Agance
Nom Agance

1.1 Les choix sur la fenêtre d’affichage

1.1.1 Le thème de couleur

Figure 1: Le thème du couleur choisi : différentes nuances de vert

Différentes nuances de vert sont utilisées pour mettre en exergue l’interface devant le background. Les couleurs des boutons sont mis en évidence dans l’interface par des couleurs plus saturés, et plus sombres.
Des ajouts mineurs de vert saturé et brillant indiquent les attributs sélectionnés et les zones de liste déroulante pour les choix multiples. Mais elles sont gardés à une présence minimale pour ne pas rompre avec le thème principal.

Pourquoi vert ? C’est très simple, c'est une application pour une banque, et qu'est-ce qui met davantage l'accent sur l'argent mieux que la couleur verte !

1.1.2 L’interface

La façon dont l'interface graphique de mon application est conçu est basée sur les éléments suivants:

- **Look dramatique.**
- **Minimiser les distractions.**
- **Soutenir la hiérarchie visuelle.**
- **Attirer l'attention sur les éléments colorés.**

Il est également conçu pour l'accessibilité, en tenant compte des éléments suivants:

- **Ajouter suffisamment de contraste de couleur. (voir figure 2, 3)**
- **Ne pas utiliser les couleurs seules pour créer des informations critiques.**
- **Concevoir des états de mise au point utilisables.**

- Utiliser des instructions avec des champs de formulaire et des entrées (en ajoutant des étiquettes/label à tout).
- Utiliser un balisage correct sur le contenu.
- Utiliser une grande police claire et facile à lire.
- Ajouter suffisamment d'espace entre les boutons et entre les lignes et les colonnes

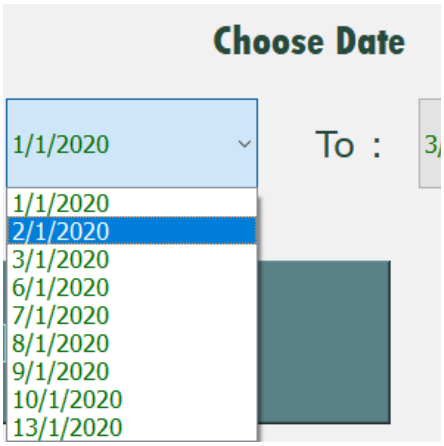


Figure 3

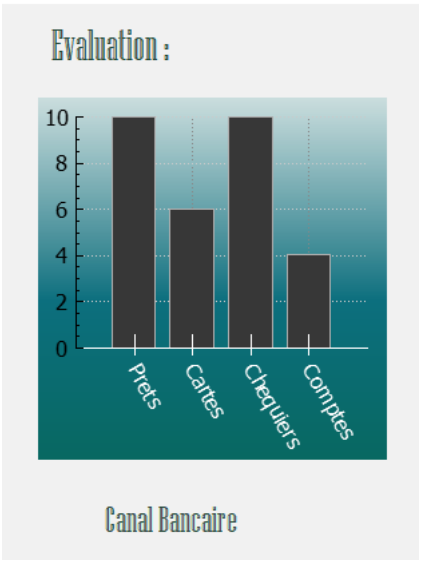


Figure 2

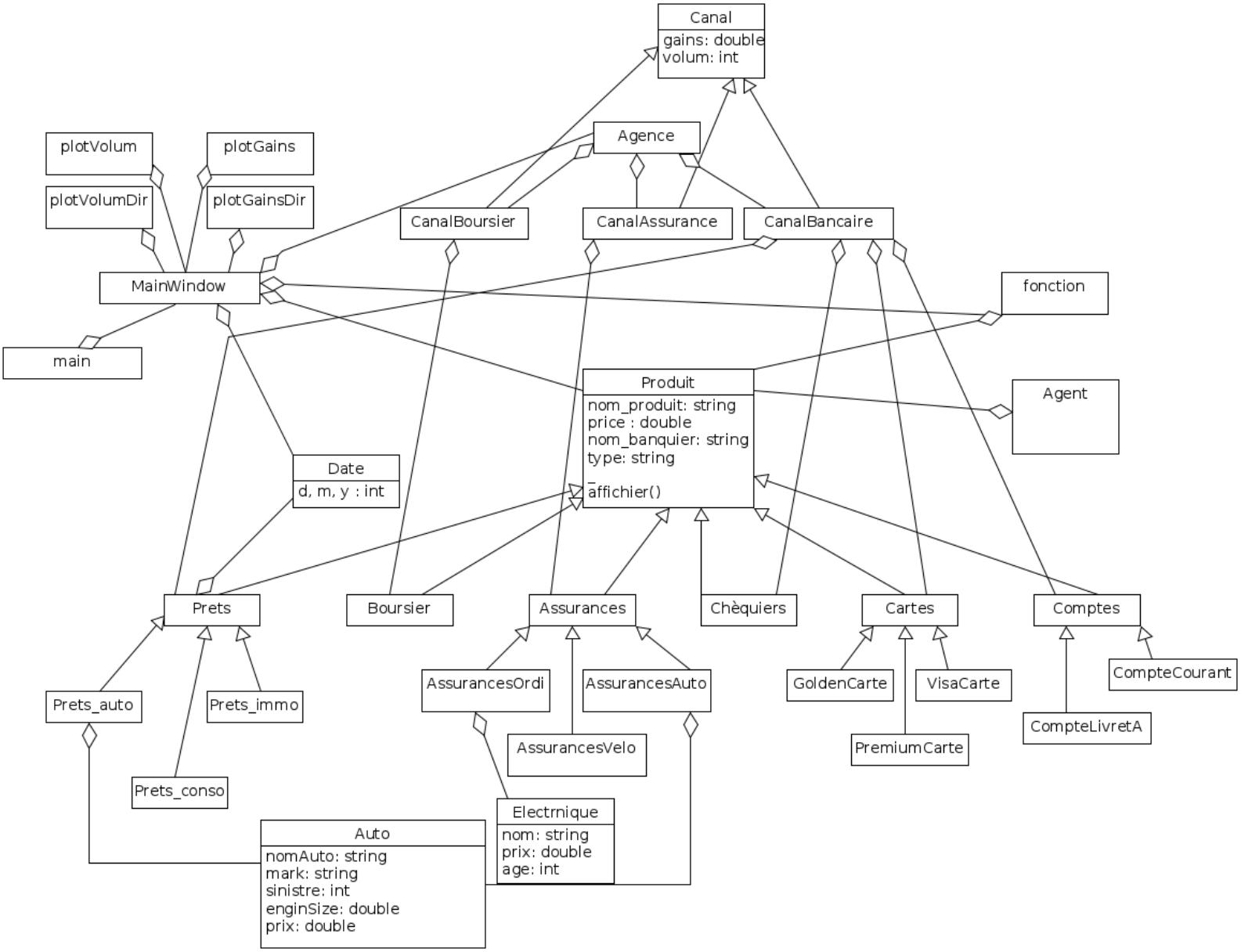
1.2.3 La résolution d’écran

Dès le début, l'application s'affiche au format paysage, pourquoi? c'est parce que plus tard, lors de la visualisation des tracés, il ne serait pas pratique de les afficher dans un format portrait. Pour conformer que le plus grand nombre d’écrans susceptibles d’utilisateur, que ce soit en mode fenêtré ou en mode plein écran, la résolution minimale par défaut est définie de 803×598.

2 Interface entre les modules

2.1 Modèle objet

Chaque Produit a sa propre classe qui hérite ses propres versions. Il existe de nombreuses classes, à commencer par la classe le grand-parent "produit" qui a ses enfants "Prêts, Comptes, Cartes, Chéquier, assurance, boursier". Chacune de ces classes a aussi ses propres enfants "Prêts Immo, ... etc." (voir le UML).


Canal
gains: double
volum: int
Agence
plotVolum
plotGains
plotVolumDir
plotGainsDir
CanalBoursier
CanalAssurance
CanalBancaire
MainWindow
fonction
main
Produit
nom_produit: string
price : double
nom_banquier: string
type: string
affichier()
Agent
Date
d, m, y : int
Prets
Boursier
Assurances
Chèquiers
Cartes
Comptes
AssurancesOrdi
AssurancesAuto
GoldenCarte
VisaCarte
CompteCourant
Prets_auto
Prets_immo
AssurancesVelo
PremiumCarte
CompteLivretA
Prets_conso
Electrnique
nom: string
prix: double
age: int
Auto
nomAuto: string
mark: string
sinistre: int
enginSize: double
prix: double

3 Choix D’implémentation:

Pour la base de données de la banque, puisqu'il n'y avait pas de type spécifique de base de données, il a été proposé que la base de données soit fournie par le client dans de simples fichiers texte, où chaque agence a son dossier de fichiers. Ces fichiers sont tous nommés en fonction de la date, par exemple : "operation01012021". Bien que ce type de base de données ne soit pas un des habituels, mais la raison pour laquelle c'était la première idée qui nous vient à l'esprit est que ce serait le plus pratique pour notre client, en tenant compte du fait qu'un banquier n'est pas un programmeur donc un fichier .text est plus susceptible d'être le type qu'il utilise.

À l'intérieur de ces fichiers, il existe de nombreux types d'opérations, chaque produit a son type d'opération unique. Pour lire cette base de données, il y a une fonction dans lz fichier "fonction" nommée lirefichier (qui prend un vecteur de produit, et un chemin / nom de fichier). Il fait un parcours dans le fichier et compare les types d'opérations aux produits, et quand il trouve une correspondance, il commence à enregistrer les éléments importants, les met dans une instance de ce produit, puis il le pousse dans ce vecteur qu'il a pris en tant que paramètre.
Toutes les autres fonctions de cette classe sont tout aussi importantes. Comme la fonction listdir (qui prend un chemin, un pointeur vers une chaîne de caractère, et un nombre) et elle fait un simple parcours du chemin donné et met les noms de ses fichiers dans le pointeur de chaîne, tandis que le nombre est pour un compteur. Cette fonction est très utile pour lire les dates de la base de données donnée (puisque la date est donnée comme partie du nom des fichiers), d'autres fonctions sont définies pour couper la date du nom et organiser les dates dans un ordre croissant.

À part le fichier « fonction », la classe agence a la deuxième fonction la plus importante. la fonction s'appelle rempli (prend un vecteur de produit), aussi simple que cela puisse paraître, elle prend un vecteur et -une fois de plus- elle compare les types à ceux que l'on veut puis elle appelle différentes fonctions des différentes classes telles que (Prêts, comptes ... etc.).

Ces fonctions sont définies pour simplement ajouter les différents types de produits (Prêts par exemple) à un vecteur qui répond à son type. Puis avec un casting dynamique, nous ajoutons ce type de produits à notre vecteur de produits. Puis on a compté les gains et/ou le volume des produits à l'intérieur de chaque canal. Puis on l’a tracé.

4.1 Difficultés rencontrés :

L'un des plus gros problèmes que j'ai eu et qui a pris du temps était de prendre le chemin de ma base de données et d'essayer de trouver une solution autre que de mettre le chemin absolu.

Un autre problème est apparu une fois que j'essayais de déclarer une variable globale. mais cela a été résolu après 3 heures de recherches.

3.Je ne savais pas qu'il y avait quelque chose appelé casting dynamique, donc j'ai continué à essayer de convertir un enfant pour être son père, et cela n'a pas fonctionné jusqu'à ce que je découvre le casting dynamique "super"

4.2 Amélioration :

- Faire un login pour le directeur et pour les employeurs .
- Finir les différentes des familles de la canal bancaire .

4.3 Bonus :

J’ai fait en sort qu’on peut faire l’évaluation par agent, j’ai implémenter les classes nécessaire pour ça et toutes les fonctionnalités mais j’ai pas eu le temps de faire la partie graphique pour cette fonctionnalité .