

RAPPORT PROJET IHM Huayi_TANG

Introduction

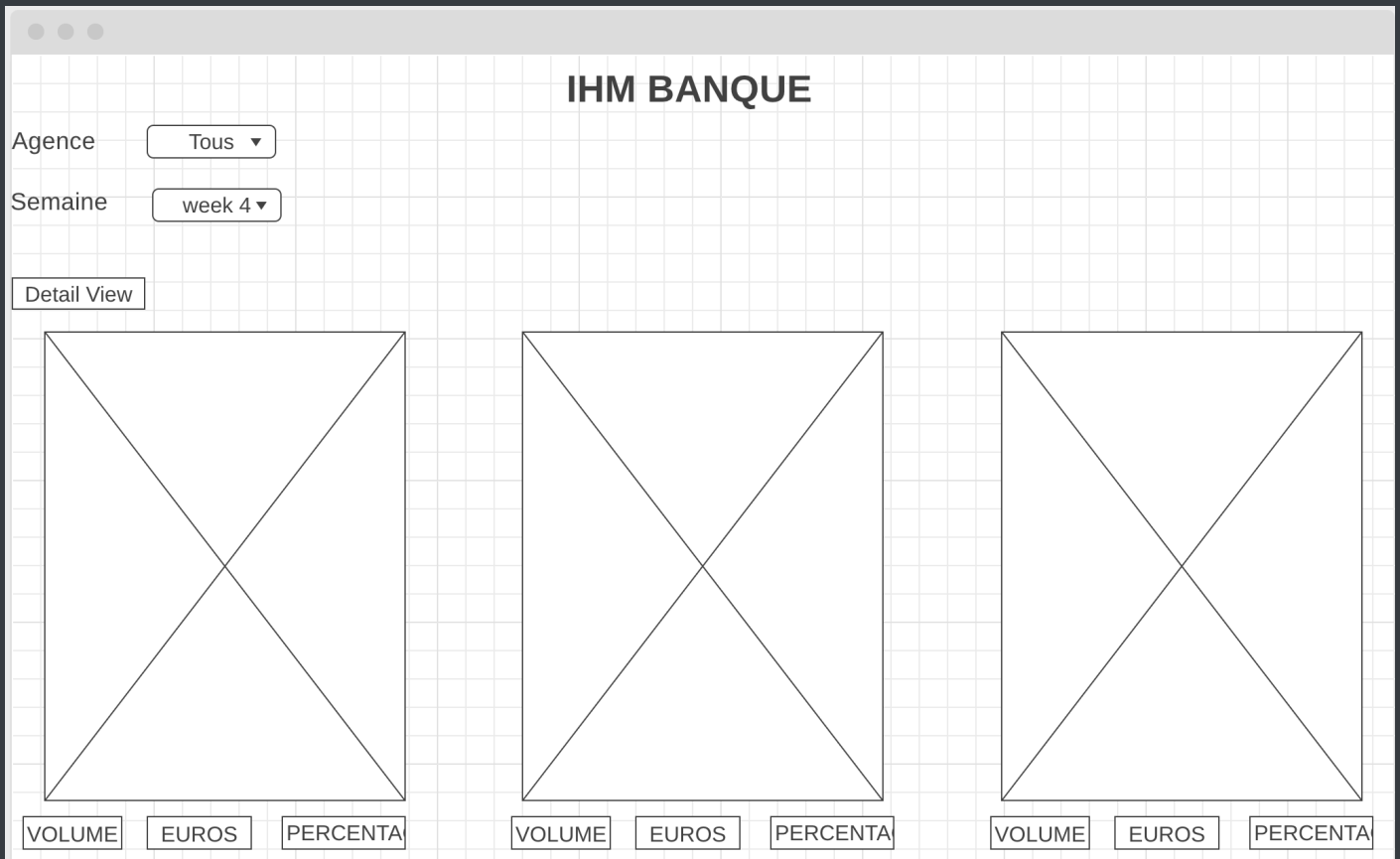
D'abord , excusez-moi de ma mal expression française et merci de votre patience:>.

Grâce aux votre conseils en cours j'ai fait cette projet étape par étape;

1. construire le base look à l'aide de QT designer
2. analyser la structure des données, et confirmer le quel type de données doit etre insérer dans la chartView
3. écrire la fonction et définir les API pour chaque classe
4. se progresser en ajouter des fonction pour chaque chartView
5. optimiser les ui design pour le rendre plus belle

WireFrame et ui Design

Inspiré par le notion "Fitts' Law" vu en cours , le design du button et comboBox est assez grand qui va facilite l'utilisateur cliquer au-dessus, et pour la mettre en page du bouton, j'ai mis les boutons en ensemble qui servent à manipuler la même chart de canal, et les combox et les bouton je les ai mis près de bord de la page ,parce que quelle que soit la distance à laquelle utilisateurs se déplace, la souris s'arrêtera finalement au bord de l'écran et sera positionnée au-dessus du bouton ou combo box.



Fonction réalisé :

1. auto initialisation du données aléatoirement en "vrai random".
2. calculer le "volume vente" et "vente en euros" par agence avec le critère de "semaine choisis " (même pour la statistique de tous les semaine)
3. calculer le "volume vente" et "vente en euros" par semaine avec le critère de "agence choisis "(même pour la statistique de tous les agences)
4. le statistic chart view de "volume de vente", "euros de vente " en BarChart, "pourcentage de volume de vente en pie chart"

//je crois que pieChart va faciliter les vue de voir "le quel type de produit vendre le plus mieux"
5. l'ordre de l'interaction sur chaque bouton est bien réalisé , c'est à dire l'ordre du choix "agence" "semaine" "volume || euro || pourcentage" n'est pas important, parce que il peut se changer fluement et correctement dans le widget.

(c'est aussi la fonction j'ai investi le plus de temps pour laisser les utilisateurs qui sens pas de "blocage" et "malaise", les view switch sont naturelle)

6. la défaut statistique view est "toutes agences" de "cette trimestre"(tous les semaine)
7. la liste dans combo box n'est pas prédéfinis, c'est a dire les éléments dans liste de combo va ajouter dynamiquement grâce à "comboBox addItem"

Fonction pas réalisé :

Je me sens vraiment désolé pour les fonctions qui sont pas réalisé, parce que le temps est pressé à cause du nombreux projets et examens de cette semestre , le délais de faire ce projet et très limité, et grâce à mon enseignant M. Pascal GUEHL(j'ai suivis tout au long le cours TP en présentiel), son cours me permet de maitriser les connaissances Qt pour la réalisation de cette projet, ça me gagner vraiment du temps en évitant apprendre Qt tout au début, merci encore.

Même si les fonctions suivantes n'est pas fait à cause du temps pressé , c'est dommage, mais j'ai les solution dans ma tête il faut juste du temps pour le réaliser, je vais le compléter après les examens pour le rendre plus belle et plus intégrale

1. La version pour chque produit n'est pas définis: ma solution est définis les class fils hérités de class père Produit pour chaque version de produit en ajoutant ses propres attributs privé .

et pour réalisé cette fonction , il faut changer l'attribut dans class Canal

```
std::vector order;
```

à

```
std::vector<Produit*> order;
```

2. Pour réaliser un "window switch" affichant les détails de version "produit prêts " définir un nouveau class "windowFils" qui hérite le mainWindow;

Dans windowFils();

- Ajouter un signal personnalisé : "backToMain"
- Ajouter un pushButton avec le slot qui va emit la signal personnalisée

"backToMain"

Dans Mainwindow:

- ajouter un nouveau attributs de type ""windowFils" de nom "windowFils"
- ajouter un slot cliquant sur le button "switch to window Fils" et qui va afficher son windowFils : windowsFils.show() et cacher le mainWindow:
- ajouter un slot pour la sigal "backToMain" de son attribut "windowFils", et il va caher le windowfils et remonter le MainWindow.

DATA STRUCTURE

Voici une bref présentation de ma conception:

J'ai divisé mon base structure de données à 3 class: Produit ,Canal, Agence

Chaque Instance Agence a 3 canal :

canal Banque (idCanal=1) ,canal Assurance(idCanal=2) canal Boursier (idCanal=3),

Chaque Instance Canal a :

idCanal: pour identifier le rôle de cette canal (canalBanque /canalAssurance/CanalBoursier)

Vector order qui va stocker tous les produit vendu;

Chaque Instance Produit contient :

int id;

(pour identifier son type; selon la quel canal il se situe,

par exemple produitId =1 est "prêt" dans canalBanque ,

mais produitId =1 est "vélo" dans canalAssurance)

```
std::string nom;
```

```
float prix;
```

```
int semaine;//entre 1~12
```

```
int volume;
```

Et pour faire la view conclusion de "tous les semaine" et "tous les agences"

j'ai mis deux cas particulier , "semaine == 0" signifie tous les semaine , "et agenceld == 0 " signifie tous les Agence

Et pour les attributs et des fonctions détaillé de chaque class, je l'ai mis à la fin pour ne pas vous embêter.

Les problèmes croisé (révision de cette projet)

1. ça me fait 2 jours de apprendre c++ pour manipuler le code, et m'adapte au conception "programmation face aux instances" en utilisant C++, donc c'est une bonne entrainement pour moi , après j'ai finis cette projet je me suis aperçus que je pourrais faire mieux et gagner plus de temps si je décidais bien la structure des class.
2. j'ai une conception pour faciliter de "créer la nouvelle agence", donc je veux faire manipuler la membre "ui" (qui est private dans la classe MainWindow) dans la constructeur de Agence.(pas encore résolu)
3. quand mon structure des données est déjà définis , je m'suis aperçu que il faut un view pour "tous les agence ensemble" et un "view pour la trimestre (12 weeks total)", donc il faut donc solution pour faire la statistique pour les deux:

est ma solution est :

Mettre un vecteur dans la classe MainWindow qui stocke tous les agences (bien sûr aussi les données des agences) pour les faire la statistique de "Tous les agences ensemble" et on le traite comme "agence0" avec `idAgence == 0` (et `idAgence` correspond "l'index du comboBox choisissant Agence"); c'est-à-dire quand la fonction calculatrice détecte le paramètre Agence dans la fonction ayant le `idAgence==0`, il va exécuter la statistique,

Et pour la statistique de tous les "tous les semaines pour chaque agences", c'est simple à traiter, la solution est juste dans la fonction de calculer toutes les semaines, il n'ajoute pas le critère (semaine indiqué) pendant la recherche, mais je dois avouer le cas "de tous les semaines de tous les agences" me prend du temps pour réparer le bug, je me suis bien réfléchi que c'est mon stratégie de structure des données qui m'induit dans ce cas difficile. j'ai appris ma leçon : c'est avant de commencer, bien penser le problème et décider la bonne structure de données pour éviter des problèmes épineux.

Class et fonctions définies dans ce projet est ci-dessous

```
class Produit
{
public:
    int id;
    std::string nom;
    float prix;
    int semaine;//entre 1~12
    int volume;
    Produit();
    Produit(int id, std::string nom, float prix, int semaine, int volume);
};

class Canal
{
public:
```

```

    Canal();
    Canal(int i);
    int addProduit(Produit& P);
    int getCanalId();
    std::vector<Produit> order;
private:
    int idCanal;//1 = bancaire 2 = assurance 3 = boursier
};

class Agence
{
public:
    int idAgence;//i = 0 totalView i = 1..n agenceView
    Canal canalBanque;//idCanal = 1
    Canal canalAssurance;//idCanal = 2
    Canal canalBoursier;//idCanal = 3

    Agence();
    Agence(int i);//constructeur en indiquant le idAgence
    int getIdAgence();
    float getEuro(int idProduit,int typeCanal,int semaine);//retourner
    les chiffre de vente en indiquant idProduit , typeCanal, et le semaine
    choisi de cette Agence
    int getVolume(int idProduit,int typeCanal,int semaine);//retourner
    les volume de vente en indiquant idProduit , typeCanal, et le semaine
    choisi de cette Agence
    void addProduitDansAgence(int idCanal,Produit pro);//ajouter un
    produit dans cette Agence
    void initAgence();//Agence se initialise ses 3 canal aléatoirement
};

class MainWindow : public QMainWindow
{
    Q_OBJECT
public:

```

```

MainWindow(QWidget *parent = nullptr);
~MainWindow();

float getEuroTous(int idProduit,int typeCanal,int semaine);//obtenir
chiffre de vente en indiquant idProduit , typeCanal, et le semaine de
tout Agence

int getVolumeTous(int idProduit,int typeCanal,int semaine);//obtenir
chiffre de vente en indiquant idproduit , typeCanal, et le semaine de
tout Agence

int getVolumeAllWeekPerAgence(Agence a,int typeCanal,int
idProduit);//obtenir volume de vente de en indiquant idAgence ,
typeCanal, et le idProduit de tout semaine

int getEuroAllWeekPerAgence(Agence a,int typeCanal,int
idProduit);//obtenir chiffre de vente en indiquant idAgence , typeCanal,
et le idProduit de tout semaine

std::vector<Agence> listAgence;//stocker les agences créer pour
faire la statistique

Agence getListAgence(int index);//pour retourner l'agence
correspondant dans la liste agence en indiquant leur idAgence, sert à
correspondre l'index change du comboBox

void paintBarVolumeCanal1(Agence agen,int idCanal,int
week);//dessiner le BarChart de volume de vente de CanalBanque en
indiquant idAgence , typeCanal, et le semaine choisi

void paintBarVolumeCanal2(Agence agen,int idCanal,int
week);//dessiner le BarChart de volume de vente de CanalAssurance en
indiquant idAgence , typeCanal, et le semaine choisi

void paintBarVolumeCanal3(Agence agen,int idCanal,int
week);//dessiner le BarChart de volume de vente de CanalBoursier en
indiquant idAgence , typeCanal, et le semaine choisi

void paintBarEuroCanal1(Agence agen,int idCanal,int week);
void paintBarEuroCanal2(Agence agen,int idCanal,int week);
void paintBarEuroCanal3(Agence agen,int idCanal,int week);
void paintPieVolumeCanal1(Agence agen,int idCanal,int week);
void paintPieVolumeCanal2(Agence agen,int idCanal,int week);
void paintPieVolumeCanal3(Agence agen,int idCanal,int week);

```



```
private slots:
    // void on_SwitchTo_clicked();
    void on_comboBoxAgence_currentIndexChanged(int index);
    void on_comboBoxSemaine_currentIndexChanged(int index);
    void on_euro_clicked();
    void on_volume_clicked();
    void on_volume_2_clicked();
    void on_volume_3_clicked();
    void on_euro_2_released();
    void on_euro_3_released();
    void on_percentage_clicked();
    void on_percentage_2_clicked();
    void on_percentage_3_clicked();
private:
    Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
}
```