

Les fonctionnalités de l'interface

MA BANQUE

Choose Agence

Directeur

Choose Date

From : 1/1/2020 To : 1/1/2020

Valider: Par volum

Valider: Par Gains

Valider: évaluation par l'agent

in progress

Mes différents combo-boxes

MA BANQUE

Choose Agence

Agence1
Directeur
Agence1
Agence2
Agence3
Agence4
Agence5

Choose Date

From : 1/1/2020 To : 13/1/2020

Valider: Par volum

Valider: Par Gains

Valider: évaluation par l'agent

in progress

MA BANQUE

Choose Agence

Agence1

Choose Date

From : 1/1/2020 To : 13/1/2020

1/1/2020
2/1/2020
3/1/2020
6/1/2020
7/1/2020
8/1/2020
9/1/2020
10/1/2020
13/1/2020

Vali

Vali

Valider: évaluation par l'agent

in progress

MA BANQUE

Choose Agence

Agence1

Choose Date

From : 1/1/2020 To : 13/1/2020

1/1/2020
2/1/2020
3/1/2020
6/1/2020
7/1/2020
8/1/2020
9/1/2020
10/1/2020
13/1/2020

Valider: Par volum

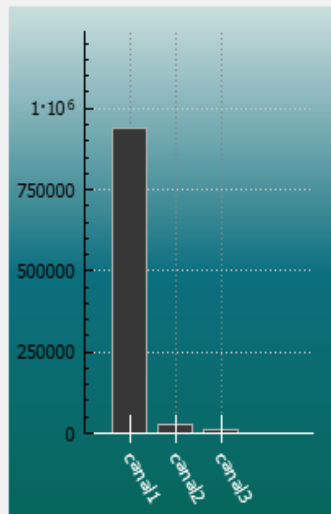
Valider: Par Gains

Valider: évaluation par l'agent

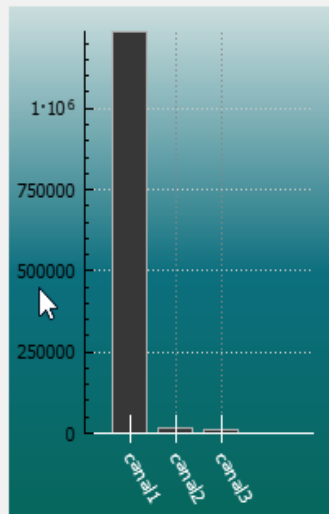
in progress

La vue du directeur

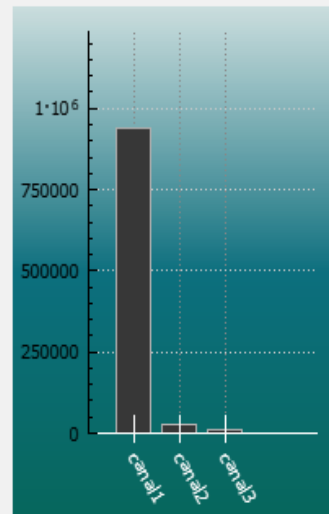
Evaluation



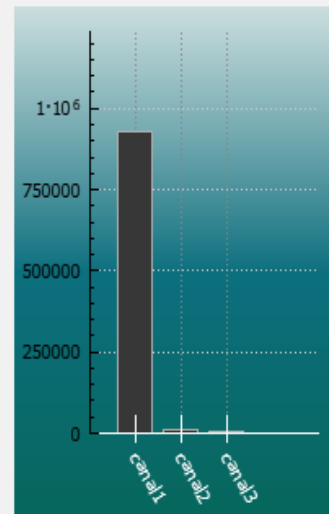
Agence1



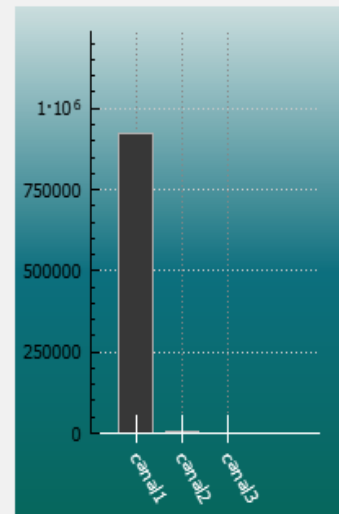
Agence2



Agence3



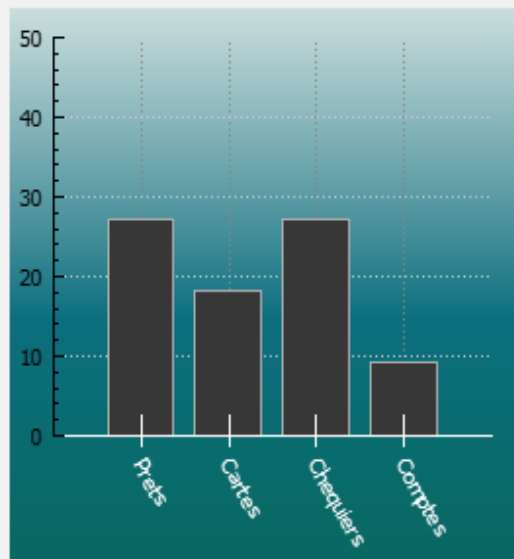
Agence4



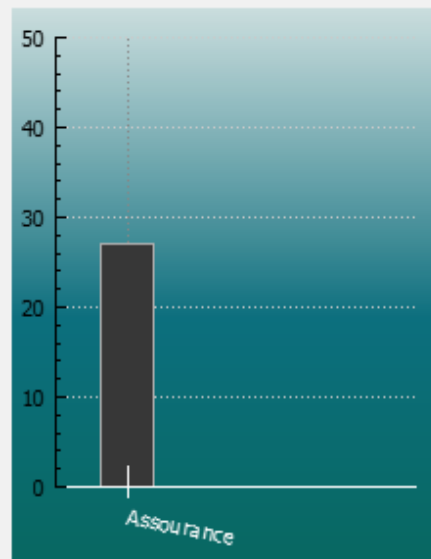
Agence5

La vue d'un agence

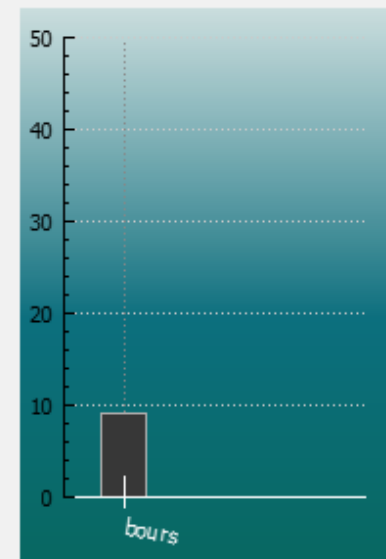
Evaluation :



Canal Bancaire

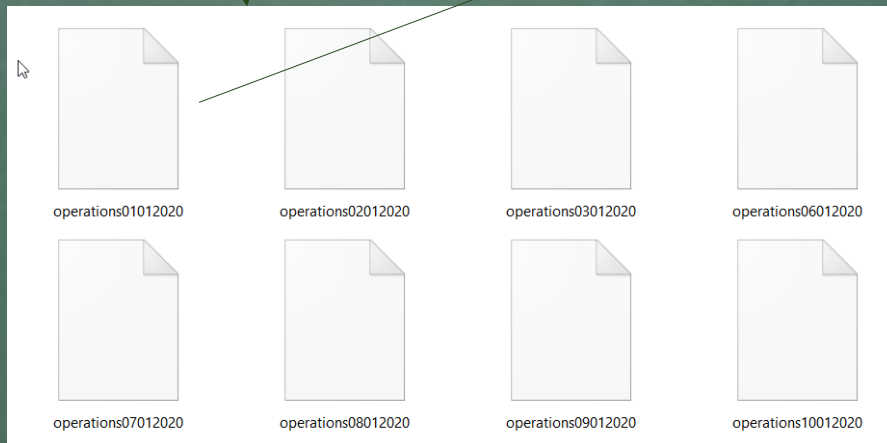
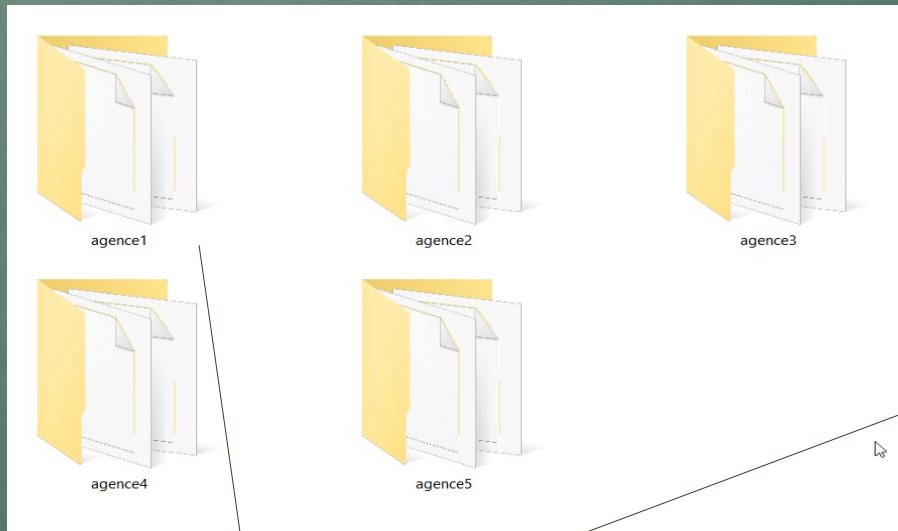


Canal Assurance



Canal Boursier

À quoi ressemble ma base de données



```
operations01012020 X
C: > Users > sundu > Desktop > ihm > project > database > agence1 > operations01012020
1  Operation: 1
2  Agent: Jean
3  Produit: Prets Immobilier
4  Nom: Prets265
5  Prix: 300000
6  Interet: 1.5%
7  Adresse: 13 rue de paris
8  Date: 02/05/2020
9
10 Operation: 2
11 Agent: Timothy
12 Produit: Assurance Auto
13 Nom: Assurance3645
14 Prix: 100
15 Duree: 3 Ans
16 NomAuto: Voiture
17 Mark: Citroen
18 Prix: 40000
19 Sinistre: 5
20 EngineSize: 1.5 L
21
22 Operation: 3
23 Agent: Jean
24 Produit: VisaCarte
25 Nom: CB3525
26 Prix: 0
27 Plafond: 450
28 Frais: 0
29
30 Operation: 4
31 Agent: Jean
32 Produit: Chequiers
33 Nom: Chequiers1225
34 Prix: 0
35 Plafond: 1500
```

Fonctions 1

```
void on_validAgence_clicked();
```

```
void function(const string pathAbs, int ind);
```

```
void lirefichier(vector<Produit*> &produitsAg1, string file);
```

```
void rempli(vector<Produit*> &v);
```

```
void addassurance(Assurances *a);
```

```
void addboursier(Boursier *b);
```

```
void ag1();
```

```
void ag2();
```

```
void ag3();
```

```
void ag4();
```

```
void ag5();
```

```
void addprets(Prets *p);
```

```
void addcomptes(Comptes *c);
```

```
void addchequiers(Chequiers *ch);
```

```
void addcartes(Cartes *crt);
```

"Function"

```
void MainWindow::function( string pathAbs,int ind){
    string path, toString;
    for (int i=0;i<=4 ;i++ ) {
        produitsAg[i].clear();
    }
    for (int i=st;i<=fi;i++)
    {
        for(int j=0; j<5; j++)
        {
            toString = to_string(j+1);
            path = pathAbs + "agence" + toString + "/" + fichiers[arrLi[i]];
            lirefichier(produitsAg[j],path);
            cout<<"Finished Reading Agence" << toString << " / " << path <<endl;
        }
    }
    switch (ind)
    {
        case 0:
            for(int i=0;i<5;i++) agence[i].rempli(produitsAg[i]);
            // puis plot le
            break;
        case 1:
            agence[0].rempli(produitsAg[0]);

            // puis plot le
            break;
        case 2:
            agence[1].rempli(produitsAg[1]);

            // puis plot le
            break;
        case 3:
            agence[2].rempli(produitsAg[2]);

            // puis plot le
            break;
        case 4:
            agence[3].rempli(produitsAg[3]);

            // puis plot le
            break;
        case 5:
            agence[4].rempli(produitsAg[4]);

            // puis plot le
            break;
        default:
            cout<<"Unexpected but strange yet fantastic Error"<<endl;
            abort();
    }
}
```


“lirefichier”

```
void lirefichier(vector<Produit*> &produitsAg1, string file)
{
    ifstream fich(file);
    string junks, agent, type, nom, infos1, infos2, infos3, infos4;
    double infod1, infod2, infod3, prix; std::ostringstream
    int junki, infoi1, infoi2;
    while(true)
    {
        fich>>junks>>junki;
        //cout<<junks<<" : "<<junki<<endl;
        if(fich.eof()) break;
        fich>>junks>>agent;
        getline(fich, junks);
        getline(fich, junks);
        type=junks.substr(9, sizeof(junks)-9);
        fich>>junks>>nom;
        fich>>junks>>prix;
        //////////*bancaire*//////////

        //////////*prets*//////////
        if (type=="Prets Immobilier")
        {
            fich>>junks>>infod1; //interet
            getline(fich, junks); // jump
            getline(fich, junks); // adresse
            infos1=junks.substr(9, sizeof(junks)-9); //adresse
            fich>>junks>>infos2; //date
            Date d(stoi(infos2.substr(0,2)),stoi(infos2.substr(3,2)),stoi(infos2.substr(6,4)));

            produitsAg1.push_back(new Prets_immo(nom,prix,agent,d,infod1,infos1));
        }
        else if (type == "Prets Consommation")
        {
            fich >> junks >> infod1 >>junks; //interet
            fich >> junks >> infos1; //date
            Date d(stoi(infos1.substr(0, 2)), stoi(infos1.substr(3, 2)), stoi(infos1.substr(6, 4)));
            fich >> junks >> infos2;
            produitsAg1.push_back(new Prets_conso(nom, prix, agent, d, infod1, infos2));
        }
    }
}
```


Fonctions 2

```
void list_dir(const char *path, string *fich, int &cnt)
{
    struct dirent *entry;
    string junk;

    DIR *dir = opendir(path);
    if (dir == NULL)
    {
        return;
    }
    cnt=0;

    while ((entry = readdir(dir)) != NULL)
    {
        junk=entry->d_name;
        if (junk[0]!='.') continue;
        fich[cnt]= entry->d_name;
        cnt++;
    }

    closedir(dir);
}
```

```
virtual void calcgains()=0;
double getgains();
int getVolum();
virtual void clear()=0;
```


```
void directeur::ag1(){
    // set dark background gradient:
    QLinearGradient gradient(0, 0, 0, 400);
    gradient.setColorAt(0, QColor(200, 220, 220));
    gradient.setColorAt(0.38, QColor(10, 105, 120));
    gradient.setColorAt(1, QColor(1, 90, 60));
    ui->ag1->setBackground(QBrush(gradient));
    QCPBars *bars = new QCPBars(ui->ag1->xAxis, ui->ag1->yAxis);
    bars->setAntialiased(false);
    //bars->setStackingGap(4);
    bars->setPen(QPen(QColor(100, 100, 100).lighter(170)));
    bars->setBrush(QColor(50, 50, 50));
    // prepare x axis with country labels:
    QVector<double> ticks;
    QVector<QString> labels;
    ticks << 1 << 2 << 3;
    labels << "canal1" << "canal2" << "canal3" ;
    QSharedPointer<QCPAxisTickerText> textTicker(new QCPAxisTickerText);
    textTicker->addTicks(ticks, labels);
    ui->ag1->xAxis->setTicker(textTicker);
    ui->ag1->xAxis->setTickLabelRotation(60);
    ui->ag1->xAxis->setSubTicks(false);
    ui->ag1->xAxis->setTickLength(10, 5);
    ui->ag1->xAxis->setRange(0, 0);
    ui->ag1->xAxis->setBasePen(QPen(Qt::white));
    ui->ag1->xAxis->setTickPen(QPen(Qt::white));
    ui->ag1->xAxis->grid()->setVisible(true);
    ui->ag1->xAxis->grid()->setPen(QPen(QColor(130, 130, 130), 0, Qt::DotLine));
    ui->ag1->xAxis->setTickLabelColor(Qt::white);
    ui->ag1->xAxis->setLabelColor(Qt::white);
    ui->ag1->yAxis->setRange(0, ymax);
    // Add data:
    QVector<double> barsData;
    barsData << _tab[0][0]+_tab[0][1]+_tab[0][2]+_tab[0][3] << _tab[0][4] << _tab[0][5];
    bars->setData(ticks, barsData);
}
```

les fonctionnalités de l'interface

- l'utilisateur commence par ajouter une date de début et une date de fin (ces valeurs sont met par défaut au premier date dans le base de donnée). il choisit alors l'agence qu'il veut évaluer (aussi par défaut met au directeur générale), puis il peut simplement voir l'évaluation de cette agence soit par ses gains de chaque canal, soit par nombre d'opération de ces canal.

Ma base de données

voilà ma base de donnée, c'est un dossier avec
de sous dossiers· un dossier par agence· puis
dans les sous dossier il y a les fichiers crée et
nommée par le dates des opérations



Fonctionnalité

- Une fois cliqué le Bouton valider, la fonction "function" fait un boucle de la fonction "lire fichier", où elle lit tous les fichiers "à partir de la première date choisie jusqu'à la dernière date choisie" de toutes les agences ,et met les informations nécessaires de chaque agence séparément dans un vecteur de produits, puis il appelle une autre fonction appelée «rempli» dans un switch, en réponse à l'index choisi dans la comboBox «choisir une agence».
- la fonction "rempli" prend un vecteur de produits, compare les types de produits qui sont à l'intérieur de ce vecteur et appelle la fonction responsive à chaque type. (addPrets pour les prêts, addCartes pour les cartes, ... etc.)
- après cela, on appelle les fonctions qui font le traçage