

Projet CDAA

Introduction et objectifs :

Le projet de CDAA consiste à créer une application qui sert à la gestion et l'entreposage de contacts, l'utilisateur doit pouvoir créer des contacts afin de stocker leurs informations et entretenir une relation avec ces mêmes contacts en attribuant des tâches à faire ou bien de simples notes.

L'utilisateur doit pouvoir naviguer dans la sa liste de contact et interagir avec ces contacts en effectuant certaines action comme afficher leurs informations, les modifier, les supprimer, leurs ajouter des interactions ainsi que des tâches, supprimer ces interactions, afficher la liste de toutes les interactions, effectuer une recherche de contact à l'aide de mot clés et critères de recherche ainsi que par intervalles de date mais aussi une recherche d'interaction à l'aide de mot clés et critères de recherche ainsi que par intervalles de date.

Deux fonctionnalités supplémentaires viennent s'ajouter à cela , l'exportation des fichiers JSON ainsi que l'affichage des Logs,

les Logs vont permettre à l'utilisateur de voir les actions effectuées depuis la première utilisation de l'application, ainsi si il pense qu'il y a quelque chose d'inhabituel il pourras consulter cet historique.

Toutes données transmise à l'application seront stockées dans une base de donnée qui se chargera de réaliser les requêtes nécessaires au besoin de l'utilisateur.

Nous verrons donc dans un premier temps l'interface graphique de notre application plus en détail, dans un second temps nous verront la structure qui nous permet la programmation orientée objet, et pour finir nous verrons les tables utilisées pour la gestion de la base de donnée.

I / Interface Graphique

L'objectif de la conception de cette application est de rendre son utilisation fluide , d'amoindrir sa complexité et donc de facilité la tâche à l'utilisateur et éviter qu'il se perde dans l'application, pour ce faire on opte pour un Widget principale nommé PrincipaleWidget qui prendra place dans notre MainWindow durant l'entièreté de l'utilisation de l'application.

IMPORTANT :

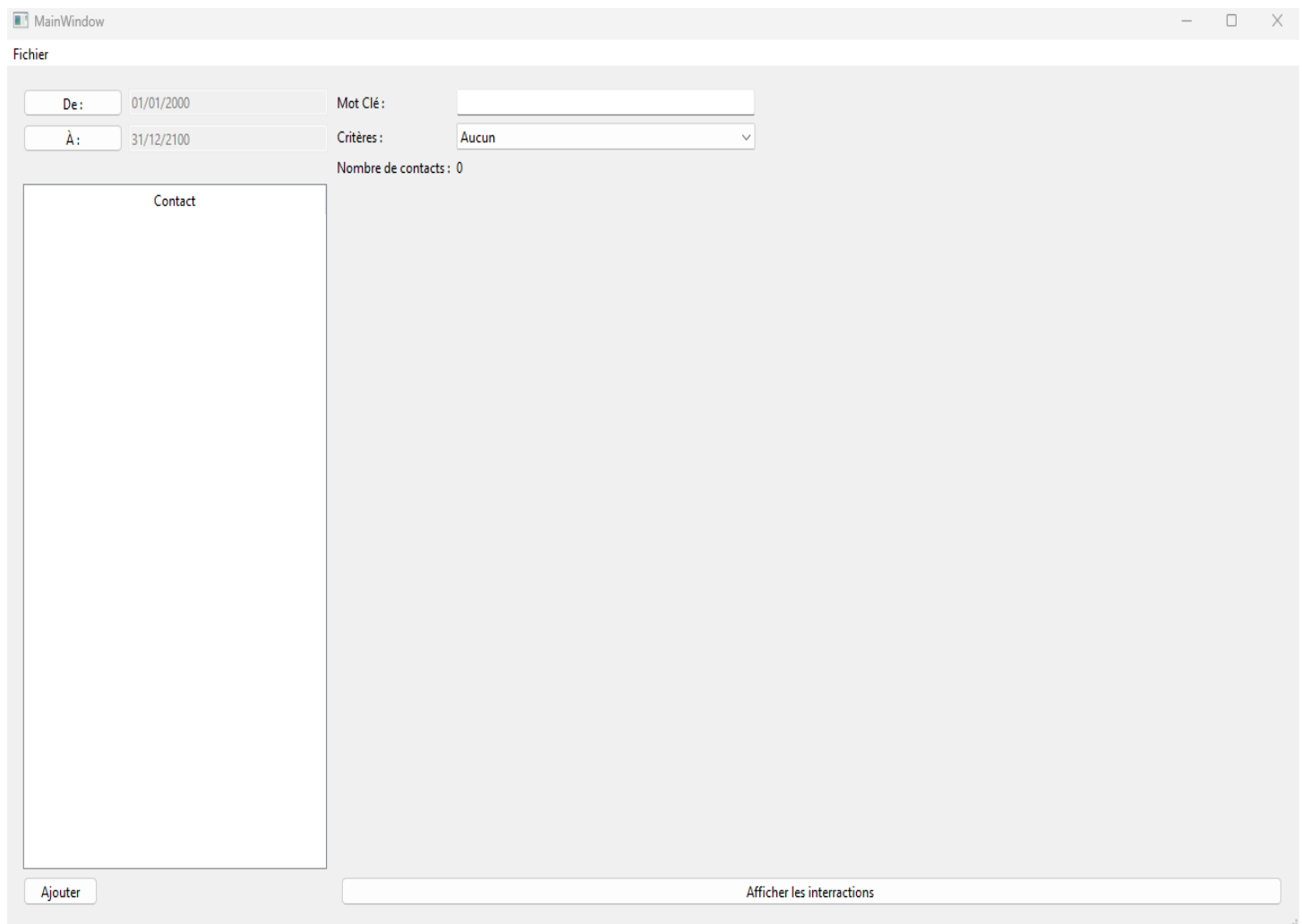
Le projet QT est réalisé avec le dernière version (6.7.0) de QT Creator, le chargé de TP nous a dit qu'il suffisait de le mentionner dans le rapport.

Le Build est ceci dit présent de le répertoire de projet.

1) PrincipaleWidget

Le Widget principale est un Widget qui façonne l'application à l'image des applications modernes tel que Skype, Discord..

L'interface permet à l'utilisateur d'accéder en permanence quelque soit l'action réalisée à la liste de contact ainsi qu'aux méthodes de recherches liées à cette liste tout en laissant place à une grande fenêtre à l'intérieur de ce même Widget destinée aux Widgets enfants.



On aperçoit sur la gauche la liste de contact actuellement vide, en dessous de celle-ci un bouton « Ajouter » pour afficher l'interface d'ajout de contact, sur la droite un bouton « Afficher les interactions » afin d'afficher la liste des interactions de tous les contacts (qui est initialement affichée au début de l'application, pour cette capture nous l'avons désactivée pour y aller progressivement), et enfin au dessus de l'application les méthodes de recherche par intervalle de date qui s'effectueront à l'aide de QCalendar et la recherche par mots clés et critère ainsi qu'un affichage du nombre actuel de contacts.

Un grand espace vide est laissé au milieu de l'application, c'est à cet espace que les Widgets enfants vont venir s'afficher.

L'utilisateur va procéder à l'ajout d'un contact donc celui-ci va appuyer sur le bouton « Ajouter », ce bouton va envoyer un signal pour déclencher un slot qui affichera le Widget Ajout Contact et fermera toute autre fenêtre susceptible d'être ouverte avant.

Ajouter

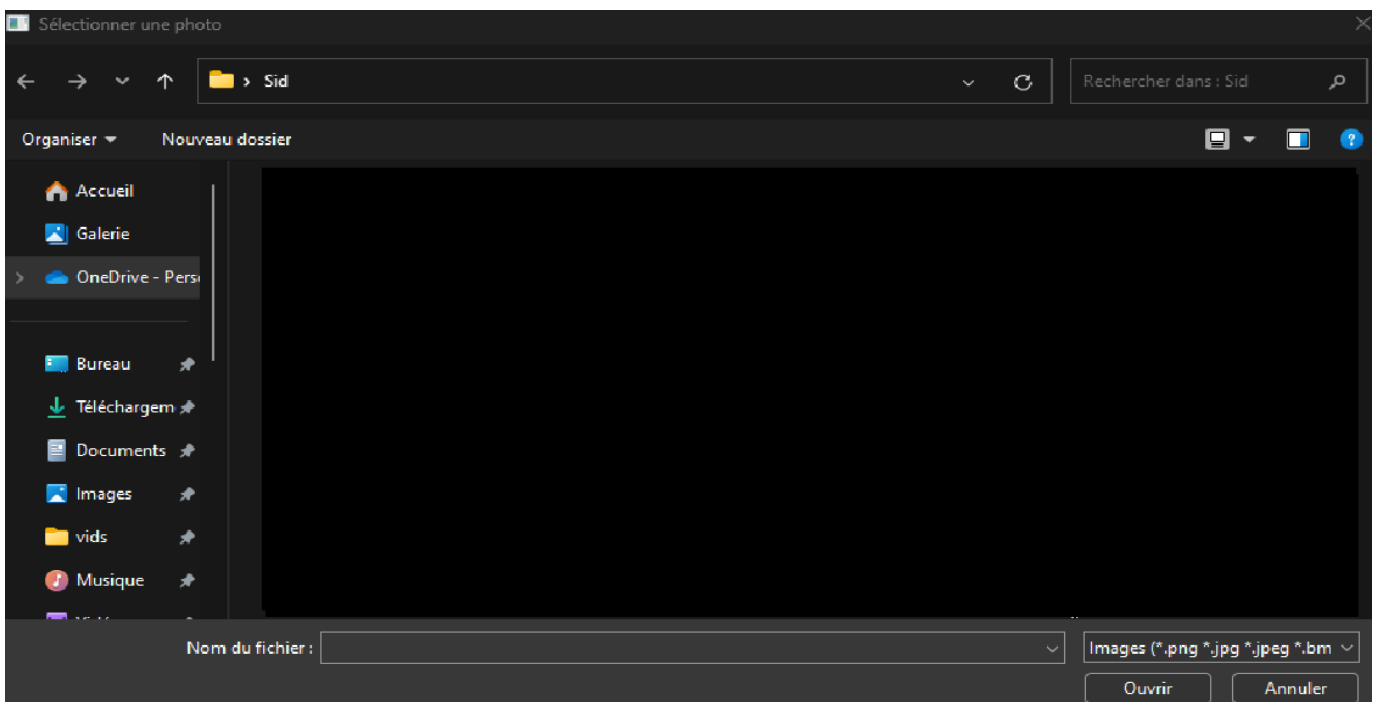
2) Ajout de contact

The screenshot shows a window titled 'MainWindow' with a menu bar containing 'Fichier'. The window is divided into several sections. On the left, there is a 'Contact' sidebar. The main area contains a form for adding a contact. At the top, there are date pickers for 'De:' (01/01/2000) and 'À:' (31/12/2100), a 'Mot Clé:' text field, and a 'Critères:' dropdown menu set to 'Aucun'. Below these, it says 'Nombre de contacts : 0'. The central part of the form is a large red rectangle with a button labeled 'Télécharger une photo'. Below this, there are input fields for 'Nom:', 'Prenom:', 'Entreprise:', 'Mail:', 'Telephone:', and 'Date:'. At the bottom of the form are two buttons: 'Annuler' and 'Valider'. At the very bottom of the window, there is a button labeled 'Ajouter' on the left and a text field labeled 'Afficher les interactions' on the right.

Voici donc notre interface désormais, l'utilisateur est prêt à entrer les informations dans l'application.

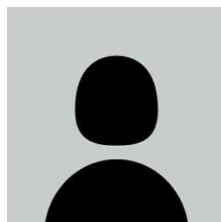
Le Widget présente donc un QLineEdit ainsi que son QLabel respectif par attribut du contact, il présente un bouton « Télécharger une photo », un bouton « Annuler » et un bouton « Valider »

L'utilisateur procède donc au remplissage du formulaire, lorsque celui ci clique sur le bouton « Télécharger une photo », un gestionnaire de fichier s'ouvre afin que celui ci sélectionne la photo.



4 types d'images sont acceptées, une fois la photo sélectionnée le Widget actualisera la photo pour donner un visuel à l'utilisateur, voici donc l'affichage après l'ajout de la photo ainsi que les informations.

à noter que si l'utilisateur n'entre pas de photo , on attribut une photo par défaut au contact qui est celle-ci



Si l'utilisateur décide d'annuler le formulaire en appuyant sur le bouton « Annuler » alors le formulaire vas entièrement se nettoyer et l'utilisateur pourra de nouveau entrer ses informations.

Dans le cas ou l'utilisateur valide en appuyant sur le bouton « Valider », alors commence plusieurs étapes de vérification du formulaire , une première étape vérifie si l'entrée est vide , une seconde vérifie si le format est correct , une troisième vérifie si le mail est déjà dans la base de donnée utilisée par un autre contact.

voici un exemple des erreurs perçues dans le cas de l'entrée mail :

Mail : Veuillez remplir cette case.	Mail : Le format est incorrect.	Mail : Le Mail est déjà existant.
<input type="text"/>	<input type="text" value="Blablabla"/>	<input type="text" value="Sid. @homtail.com"/>

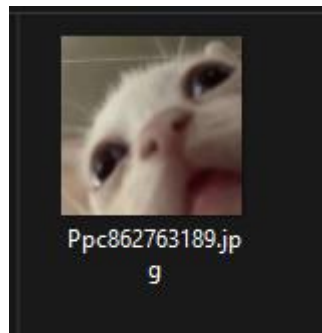
On a donc un regex qui vérifie les entrées basiques tel que le nom, prenom , entreprise qui sont une chaîne de caractère de l'alphabet contenant des espaces et des tirets.

un regex pour le mail qui vérifie ce genre de format : bla12.bla12@bla12.bla et bla12@bla12.bla

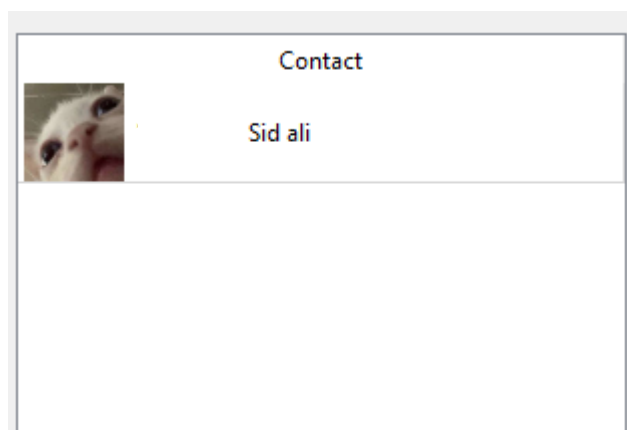
et un dernier regex qui vérifie le format de téléphone français c'est à dire le +33 ou le 0 suivis de 9 chiffres.

L'entrée Date est grisée car la date du jour de la création du contact est attribuée directement au contact au moment de sa création.

Si il n'y a pas d'erreurs alors on valide la création du contact ,
on appelle alors une méthode qui va récupérer l'URL de la photo et copier cette photo dans un répertoire de
l'application prévu au stockage des photos de profil (..\GestionContact\Assets\PhotoContact)
Le Widget va par la même occasion changer le nom de cette photo en générant un nombre aléatoire de 7 à 8
chiffres et en l'affectant au nom de l'image avec le préfixe "Ppc".



Le Widget finit par émettre un signal au Widget principale contenant le contact qui enverra de nouveau un
signal à la base de données avec le contact à l'aide d'un slot qui par la même occasion fermera le Widget
AjoutContact.
Après avoir lu la requête , la base de données renvoie un signal au Widget principale pour rafraîchir la liste de
contact.



La liste de contact se présente donc comme ceci et laisse apparaître le nom et prénom du contact en plus de
sa photo de profil.

3) Recherche de contacts

Une fois que la liste possède plusieurs contacts , l'utilisateur va pouvoir trouver un intérêt à la recherche de contact. Dans une première partie, nous avons la recherche par intervalle de date. Et dans une 2nde partie, la recherche par mot-clé et par critère.

De :	01/01/2000	Mot Clé :	
À :	31/12/2100	Critères :	Aucun
Nombre de contacts : 2			

Dans le cas de l'intervalle de date, l'utilisateur a accès à 2 boutons qui sont parallèlement alignés a un QLineEdit chacun, le bouton du dessus « De : » fait office de date A. Et le bouton du dessous « À » fait office de dateB, l'intervalle de recherche se déroule donc de la date A à la date B. Les dates sélectionnées par l'utilisateur seront actualisées dans les QLineEdit.

De :	01/01/2000
À :	31/12/2100

Afin que l'utilisateur sélectionne une date, Il lui faut cliquer sur le bouton « De : » Pour la date A ce qui aura pour effet d'ouvrir un QCalendar, l'utilisateur vas ensuite choisir une date dans le calendrier et le QLineEdit sera mis à jour.

GestionContact

décembre, 2023

	lun.	mar.	mer.	jeu.	ven.	sam.	dim.
48	27	28	29	30	1	2	3
49	4	5	6	7	8	9	10
50	11	12	13	14	15	16	17
51	18	19	20	21	22	23	24
52	25	26	27	28	29	30	31
1	1	2	3	4	5	6	7

Ici on sélectionne la date du 22 décembre 2023

De :	22/12/2023
------	------------

QLineEdit après avoir sélectionné la date.

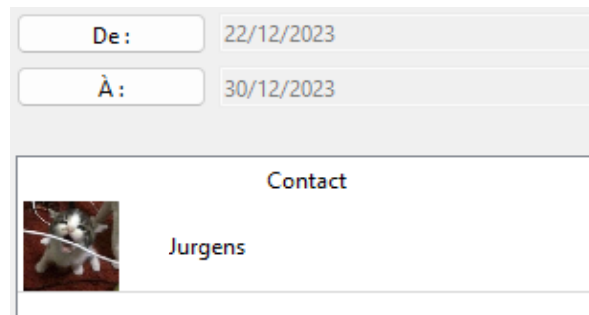
Le principe est le même pour le bouton de la date B.

À noter que l'utilisateur ne peut pas sélectionner une date B inférieure à la date A, et si celui-ci sélectionne une date A supérieure à la date B, alors la date B prendra la valeur de la date A, ce qui résulte à avoir un intervalle dans le même jour.

Lorsqu'une date est sélectionnée et donc le QLineEdit actualisé Alors un signal était émis à l'intérieur du widget principal. Afin de rafraîchir la liste de contact.

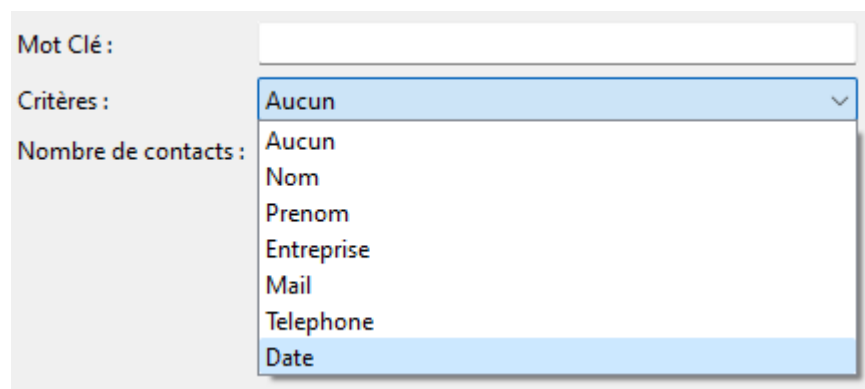


Voici mes deux contact , le premier a sa date de création le 20/12/2023 et le second a sa date le 22/12/2023




Voilà l’affichage résultant à la recherche.

Dans le cas de la recherche par mots-clés et critères l'utilisateur devra sélectionner un critère et parmi ces critères il y a le critère par défaut qui est "aucun" et il y a ensuite les critères "non" "prénom" "entreprises" "mail" "téléphone" et "date".

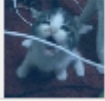


Lorsque l'utilisateur entre un texte dans la ligne de mots clés cette ligne de recherche envoie un signal au Widget principal pour rafraîchir la liste de contacts, de même lorsque l'utilisateur sélectionne un critère dans le QComboBox.

Contact



Dupont Sid ali




Jurgens

Mot Clé :

Critères :

Nombre de contacts : 2

Contact



Dupont Sid ali

Résultat de la recherche par Nom “ Dupont ”.

Lors du rafraîchissement de la liste de contacts, le Widget principal émet un signal à la BDD afin de récupérer la liste de contacts. Une fois la requête terminée, la base de données renvoie la liste de contacts au Widget principal après quoi la création Des QTablesWidgetItem contenant les contacts va commencer.


La méthode va parcourir la liste de contacts et vérifiez pour chaque contact si les critères de recherche tous compris correspondent aux contacts. Si les critères correspondent, alors on crée l'interface du contact dans la liste de contacts, si ils ne correspondent pas alors on poursuit la liste.

L'utilisateur va donc désormais cliquer sur le contact afin d'afficher la fiche de ce contact qui est le Widget Affiche Contact, La fiche de ce contact va donc prendre place dans le Widget principal.

4) Affiche Contact

Les informations du contact sur lequel l'utilisateur a cliqué vont donc s'afficher sur la fiche du contact. Par la même occasion, l'utilisateur aura accès à différents boutons comme le bouton « Modifier », « Ajout Interaction » et « Supprimer » .

L'utilisateur peut aussi percevoir un tableau qui contiendra la liste d'interactions de ce contact ainsi que 2 boutons grisés « Supprimer interaction » et « Afficher interaction » .



Sid ali

sid . @hontail.com

Etudiant

06

20/12/2023

Supprimer Interaction

Afficher Interaction

Date	Interaction	Todo

Modifier


Ajout Interaction

Supprimer

Dans un premier temps, nous allons voir le cas où l'utilisateur veut modifier le contact.

4.1) Modification Contact

Lorsque l'utilisateur clique sur le bouton « Modifier », la fiche contact se ferme et laisse place à un autre Widget du nom de Modif contact.



Télécharger une photo

Supprimer la photo

Nom :

Prenom :

Sid ali

Entreprise :

Mail :

Etudiant

sid . @hontail.com

Telephone :

Date :

06

20/12/2023

Annuler

Modifier

Très similaire au Widget Ajout Contact, le Widget est un formulaire qui va permettre à l'utilisateur de modifier le contact, lors du chargement du widget, on charge les données du contact afin de les insérer directement en entrées.

Cela facilite la modification car l'utilisateur n'a pas besoin de rentrer de nouveau les données qu'il ne veut pas changer.

Un nouveau bouton fait son apparition « Supprimer la photo », lorsque l'utilisateur clique sur ce bouton, la variable de la photo est vidée et si l'utilisateur valide la modification alors le contact prendra la photo par défaut de contact.

Si l'utilisateur clique sur le bouton « Annuler » alors le formulaire va se fermer pour laisser place de nouveau à la fiche du contact.

Si l'utilisateur clique sur le bouton « Valider » alors 3 étapes de vérification vont se lancer semblables à l'Ajout Contact.

Une vérification vérifie si les entrées sont vides, la seconde vérifie si les entrées contiennent une erreur de format. Et la 3e vérifie si le mail en entrée existe déjà dans la BD ou non. Cas d'exception si la modification n'inclut pas le changement du mail.

Si les 3 vérifications n'affichent aucune erreur, alors dans ce cas-là, le Widget Modif Contact envoie un signal contenant le contact de base que l'on veut modifier ainsi que le contact après modification vers le Widget principale. Le Widget principale enverra ensuite le signal à la base de données et fermera le formulaire de Modification Contact.

Une fois que la requête transmise à la base de données et exécutée, alors, la base de données envoie un signal pour rafraîchir la liste de contacts dans le principal widget.

4.2) Suppression Contact

Lorsque l'utilisateur souhaite supprimer le contact il clique sur le bouton « Supprimer », le Widget principale émet un signal contenant le contact actuellement sélectionné vers la base de données.

Une fois ce signal emit, la requête dans la base de données est exécutée et elle renverra un signal pour rafraîchir la liste de contacts.

4.3) Ajout Interaction

Lorsque l'utilisateur souhaite ajouter une interaction il clique sur le bouton « Ajout Interaction », La fiche contact va se fermer pour laisser place au Widget Ajout Interaction.

Ajout Interaction : [redacted] Sid ali Retour

Contenu Interaction :

Date Interaction

Contenu Interaction

Contenu Todo :

Date Todo

Contenu Todo

Ensemble des Todo :

Liste des Todo à ajouter à l'interaction

Ajouter Todo

Supprimer le dernier Todo

Annuler

Valider

Le formulaire d'ajout d'interaction sera chargé au nom du contact sélectionné, il comporte 4 entrées différentes.

Une entrée pour la date de l'interaction, une autre pour le contenu de l'interaction

Une entrée pour la date des todos, une seconde entrée pour le contenu des todos.

Lorsque l'utilisateur entre les valeurs dans les entrées des todos. Il lui faut appuyer sur le bouton « Ajouter Todo ». afin de créer le todo et le rajouter à une liste globale de la classe.

A form titled "Contenu Todo :" with a pink header. It contains two text input fields: the first is labeled "12/10/2023" and the second is labeled "Blabla". Below the inputs is a button labeled "Ajouter Todo".

A form titled "Contenu Todo :" and "Ensemble des Todo :" with a pink header. It is divided into two main sections. The left section, titled "Contenu Todo :", has two input fields labeled "Date Todo" and "Contenu Todo", and a button labeled "Ajouter Todo". The right section, titled "Ensemble des Todo :", has a single input field containing "@Todo Blabla @Date 12/10/2023" and a button labeled "Supprimer le dernier Todo".

Une fois le todo ajouté, on rafraîchit le QTextEdit sur le côté qui affichera la liste entière de todos de l'interaction et c'est cette liste qui sera ajoutée à la fin à l'interaction lors de la validation.

Si l'utilisateur souhaite supprimer un todo, celui-ci devra appuyer sur le bouton « Supprimer le dernier Todo » ce qui fera supprimer le dernier todo dans la liste de todo, cette manière de faire pose un problème car si l'utilisateur veut supprimer le *énième* todo, Il devra supprimer tous les todo qui suivent le *énième* todo.

La solution à ce problème est de créer une liste de todo tout comme la liste de contacts qui est sélectionnable ce qui faciliterait la suppression du *énième* todo.

Si l'utilisateur clique sur le bouton « Annuler », alors le formulaire sera totalement nettoyé.

Si l'utilisateur clique sur le bouton « Retour » alors le formulaire d'ajout d'interaction va se fermer pour laisser place de nouveau à la fiche du contact.

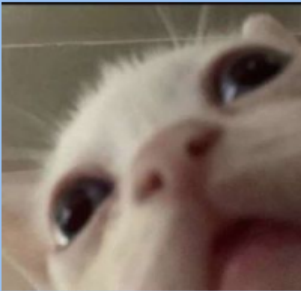
Lorsque l'utilisateur clique sur le bouton « Valider » une étape de vérification se lance afin de vérifier le format entré dans les entrées de date de l'interaction.

Une seconde vérification vérifie si le contenu de l'interaction est vide, dans le cas où des erreurs sont présentes alors des messages s'afficheront à la même manière que dans le formulaire modifier contact ou bien ajout contact.

Si le formulaire ne présente aucune erreur alors une nouvelle interaction est créée, on insère les informations du formulaire dans cette interaction afin de l'envoyer dans un signal vers le Widget principale.

Ce même signal sera envoyé à la base de données afin d'effectuer la requête d'ajout d'interactions., La BDD renverra ensuite un signal afin de rafraîchir la liste de contacts. Et de rafraîchir la fiche du contact sélectionné pour pouvoir afficher l'interaction dans la liste d'interactions de la fiche contact.

À noter que si l'utilisateur ne rentre pas de date dans les entrées de date de todo et d'interaction, Les todo et les interactions prendront la date du jour de création.



Sid ali

sid. @hontail.com

Etudiant

06

20/12/2023

Supprimer Interaction

Afficher Interaction

Date	Interaction	Todo
22/12/2023	je viens de créer une interaction	@Todo Blabla @Date 12/10/2023;


Modifier

Ajout Interaction

Supprimer

L'utilisateur se retrouve donc avec une interaction dans son contact choisi, il lui est désormais possible d'interagir avec cette interaction.

Lorsqu'il clique dessus, les 2 boutons « Supprimer Interaction » et « Afficher Interaction » vont s'activer et l'utilisateur va pouvoir manipuler l'interaction.



06

20/12/2023

Supprimer Interaction

Afficher Interaction

Date	Interaction	Todo
22/12/2023	je viens de créer une interaction	@Todo Blabla @Date 12/10/2023;

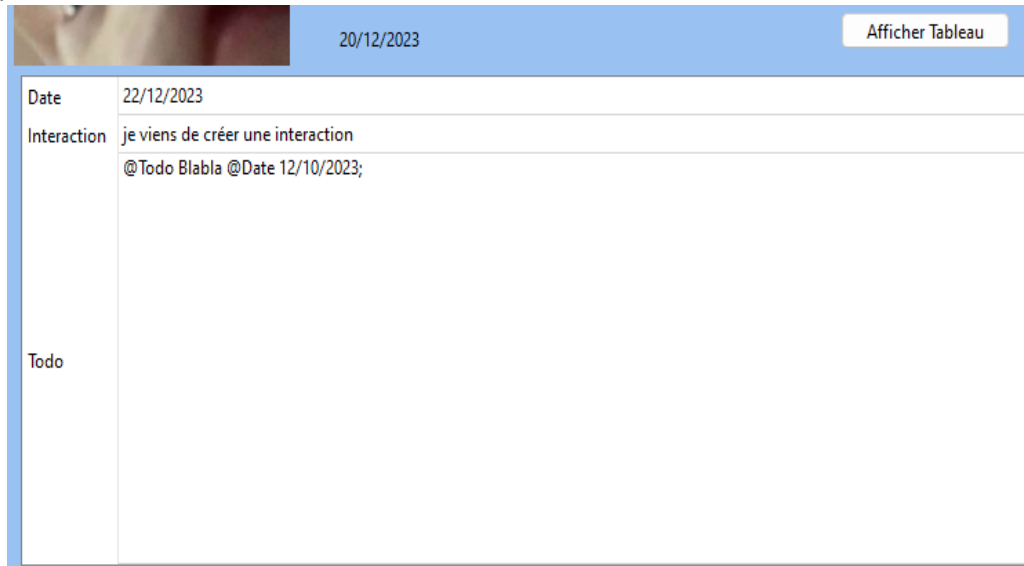
Si l'utilisateur clique sur supprimer l'interaction alors l'interaction va se supprimer de la même manière qu'un contact est supprimé, c'est à dire qu'un signal va être envoyé contenant l'interaction du contact ainsi que le mail du contact.

Le Mail étant l'ID d'un contact il est nécessaire de l'envoyer pour éviter un cas spécial :

Le cas où 2 contacts possèdent une interaction identique et que la suppression de l'interaction du mauvais contact arrive.

Une fois la requête exécutée dans la base de données elle renverra un signal vers le Widget principal afin de rafraîchir la liste de contacts et ainsi que la fiche du contact sélectionné.

Si l'utilisateur clique sur afficher l'interaction alors le tableau qui présente la liste d'interactions dans la fiche contact va changer pour laisser place à un second tableau qui lui va afficher en détail l'interaction sélectionnée en laissant plus de place aux informations de cette interaction ce qui permet une meilleure lecture de celle-ci dans le cas où elle aurait beaucoup de données.



The screenshot shows a contact detail form. At the top, there is a header bar with a date '20/12/2023' and a button labeled 'Afficher Tableau'. Below this, the form is divided into sections. The first section has a label 'Date' and a value '22/12/2023'. The second section has a label 'Interaction' and a value 'je viens de créer une interaction'. Below this, there is a text input field containing '@Todo Blabla @Date 12/10/2023;'. The third section has a label 'Todo' and a large empty text area.

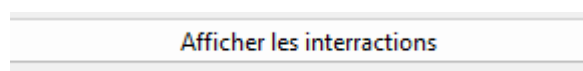
On remarque que le bouton afficher interaction a changé de nom et est désormais devenu « Afficher Tableau »

Si l'utilisateur clique sur « Afficher Tableau » alors on reviendra au tableau initial qui est l'ensemble des interactions du contact.

5) Retour au Widget principale

Désormais, revenons sur le Widget principal.

Lorsque l'utilisateur clique sur le bouton « Afficher les interactions » Toute fenêtre pouvant être ouverte à ce moment-là va se fermer pour laisser place au Widgets Liste Interactions, ainsi à l'ouverture de ce widget la liste d'interaction est rafraîchie.



The screenshot shows a button with the text 'Afficher les interactions' in a blue font, centered within a light gray rectangular frame.

Nom	Prenom	Date	Interaction	Todo
Sid ali		22/12/2023	je viens de créer une interaction	@Todo Blabla @Date 12/10/2023;

De : 01/01/2000

Mot Clé :

À : 31/12/2100

Critères : Aucun

Nombre d'Interactions : 1

On aperçoit dans ce Widget un QTableWidgetItem qui servira de liste pour toutes les interactions de tout contact et en dessous de cette liste, les méthodes de recherche similaires au Widget principal avec la liste de contacts ainsi que l'affichage du nombre d'interactions actuelles.

La liste des interactions présente 5 colonnes, le nom, le prénom, la date, l'interaction, le todo.

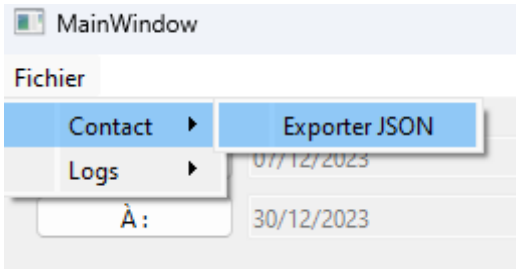
Les méthodes de recherche change seulement au niveau des critères comprenant le critère par défaut qui est "aucun" et 5 autres critères qui sont "nom contact" "prénoms contact", "date interaction", "contenu interaction" "contenu todo".

6) MainWindow

Le MainWindow est la fenêtre de notre application, en plus d’afficher le Widget principale, elle possède deux fonctions utiles à l’utilisateur.

6.1) L’exportation de contacts en fichier JSON

Dans le menu de MainWindow l’utilisateur a accès à un bouton d’action « Exporter JSON » dans uen rubrique contact.



Lorsque l'utilisateur clique sur ce bouton alors une méthode dans le MainWindow vas envoyer un signal au Widget principale pour communiquer avec la BDD et rafraichir la liste de contact, le Widget principale vas ensuite renvoyer la liste de contact au MainWindow.

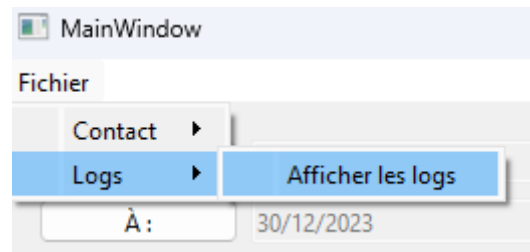
Il sera demandé à l'utilisateur de choisir un répertoire dans lequel il veut stocker son fichier du nom de GestionContact.Json .

une fois cela fait le fichier sera généré.

```
C:\Users
[
  "Contacts": [
    {
      "Date": "20/12/2023",
      "Entreprise": "Etudiant",
      "Interactions": [
        {
          "Contenu": "je viens de créer une interaction",
          "Date": "22/12/2023",
          "Todos": [
            {
              "Contenu": "Blabla",
              "Date": "12/10/2023"
            }
          ]
        }
      ]
    },
    {
      "Mail": "sid. [REDACTED]@homtail.com",
      "Nom": "Boupacha",
      "Photo": "../GestionContact/Assets/PhotoContact/Ppc862763189.jpg",
      "Prenom": "Sid ali",
      "Telephone": "06 [REDACTED]"
    },
    {
      "Date": "22/12/2023",
      "Entreprise": "Leclerc",
      "Interactions": [
        {
          "Contenu": "je viens de créer une interaction",
          "Date": "22/12/2023",
          "Todos": [
            {
              "Contenu": "Blabla",
              "Date": "12/10/2023"
            }
          ]
        }
      ]
    },
    {
      "Mail": "[REDACTED]@hotmail.com",
      "Nom": "[REDACTED]",
      "Photo": "../GestionContact/Assets/PhotoContact/Ppc856440607.jpg",
      "Prenom": "Jurgens",
      "Telephone": "+33 [REDACTED]"
    }
  ]
}
```

6.2) Affichage des Logs

La seconde fonction auquel l'utilisateur a accès est l'affichage de l'historique de l'application.



Le MainWindow envoie un signal au Widget principale pour afficher son Widget enfant “Affiche Logs”
le Widget sera chargé avec une liste de string qui contienne l’entièreté de l’historique de l’application stockée dans la base de donnée



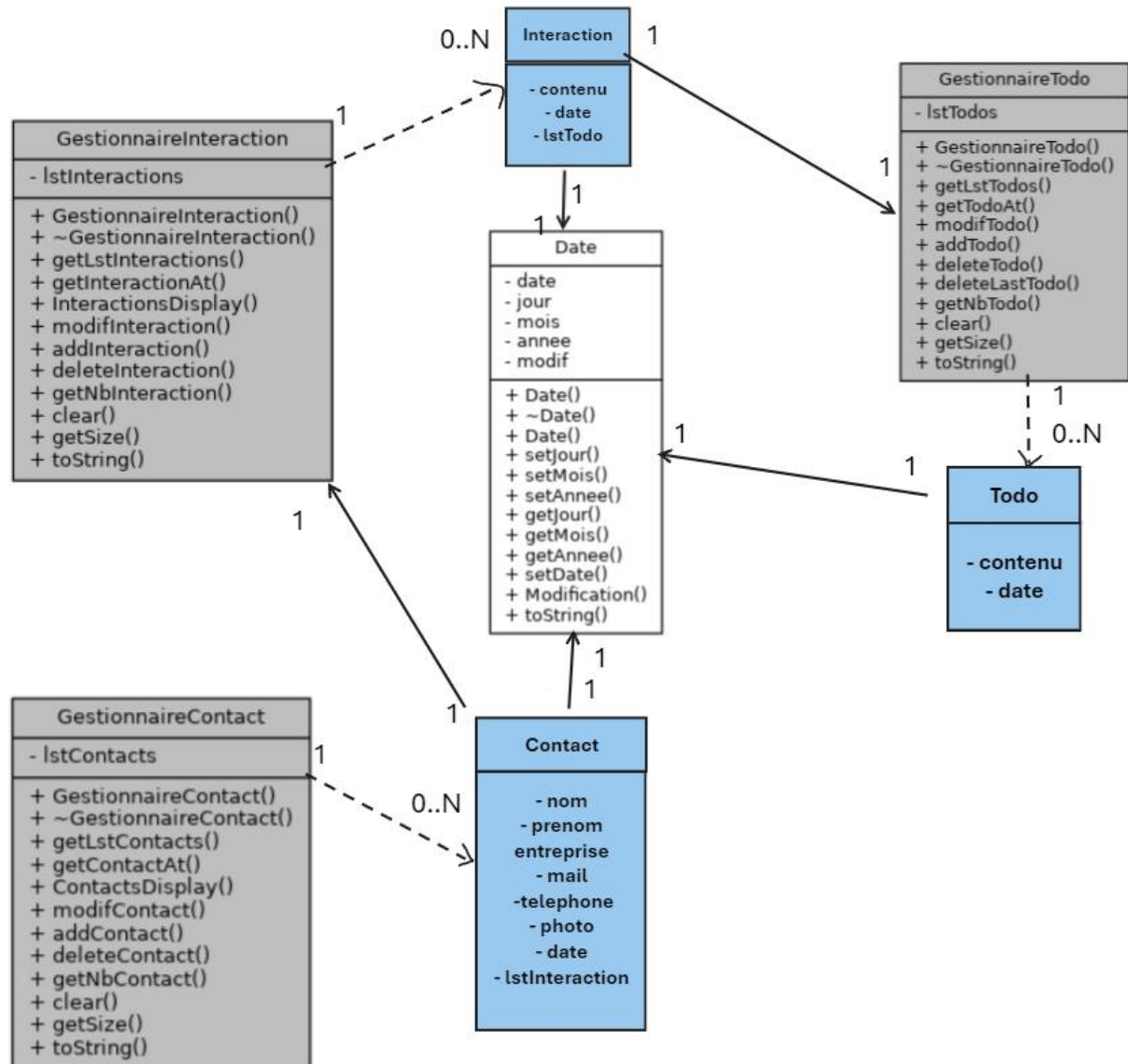
Six classes sont nécessaires à la partie de programmation orientée objet pour le fonctionnement de l'application.

Parmi elle se trouvent les classes : Contact , Interaction , Todo , Date.

Suivies des classes qui permettent leurs gestions : GestionnaireContact, GestionnaireInteraction, GestionnaireTodo.

Les classes gestionnaires ne sont ni plus ni moins que des listes de leurs type d'objets, et servent à accéder à ces listes et les manipuler pour faciliter le développement de l'application.

Voilà un diagramme de classe qui expliquerait les liens et références entre ces classes.



Vous pouvez aussi regarder dans le doxyfile.

III / Base de donnée

Pour créer notre base de donnée on insert d'abord une table Contact :
cette table contact se présente comme ça :

```
CREATE TABLE IF NOT EXISTS CONTACT(  
    "Nom TEXT NOT NULL,"  
    "Prenom TEXT NOT NULL,"  
    "Entreprise TEXT NOT NULL,"  
    "Mail TEXT PRIMARY KEY NOT NULL,"  
    "Telephone TEXT NOT NULL,"  
    "Photo TEXT,"  
    "Date TEXT NOT NULL"  
);
```

viens ensuite la table Interaction :

```
CREATE TABLE IF NOT EXISTS INTERACTION(  
    "InterID INTEGER PRIMARY KEY AUTOINCREMENT,"  
    "Cid INTEGER NOT NULL,"  
    "Contenu TEXT NOT NULL,"  
    "Date TEXT NOT NULL,"  
    "FOREIGN KEY (Cid) REFERENCES CONTACT (Mail)"  
);
```

La table todo :

```
CREATE TABLE IF NOT EXISTS TODO(  
    "TodoID INTEGER PRIMARY KEY AUTOINCREMENT,"  
    "Iid INTEGER NOT NULL,"  
    "Contenu TEXT NOT NULL,"  
    "Date TEXT NOT NULL,"  
    "FOREIGN KEY (Iid) REFERENCES INTERACTION (InterID)"  
);
```

Et pour finir la table logs :

```
CREATE TABLE IF NOT EXISTS LOGS(
    "Contenu TEXT NOT NULL,"
    "Date TEXT NOT NULL"
);
```

La table contact possède les attributs d'un contact, on définit son adresse mail comment étant l'identifiant car une adresse mail est propre a un contact.

Pour l'interaction on lui définit ses attributs comme son contenu et sa date, son ID du nom d'InterID est un integer qui sera auto incrémentée à chaque nouvelle interaction.

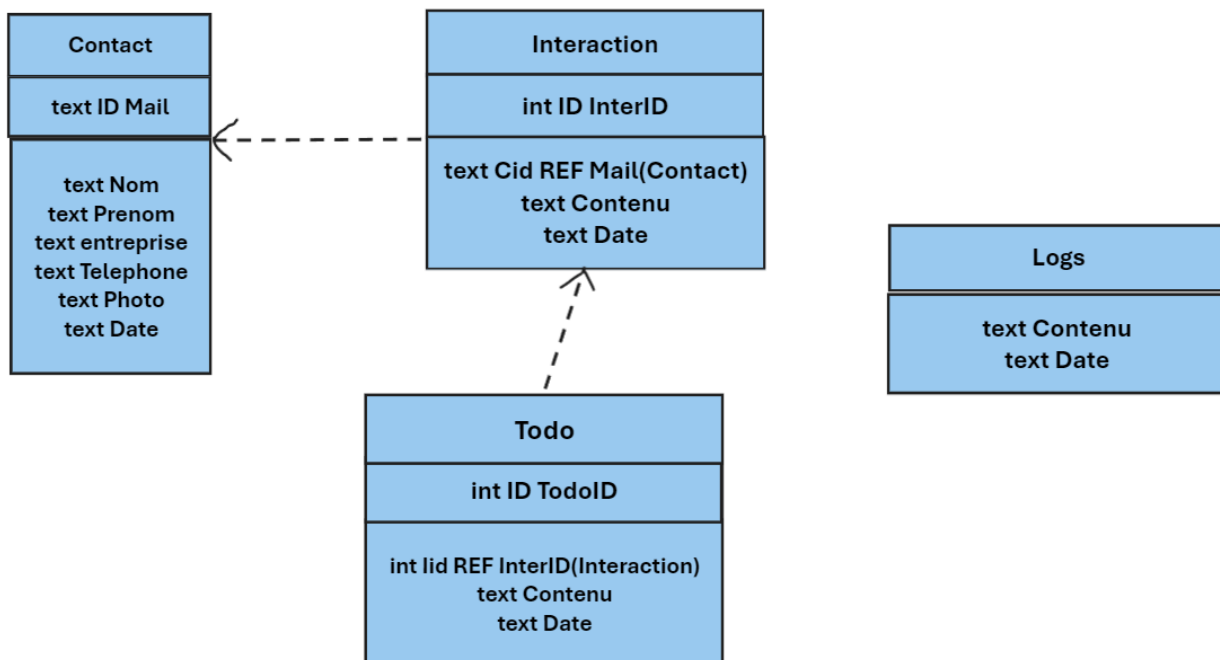
Le dernier attribut est Cid qui est une référence au mail (l'id) du contact.

Pour le Todo on lui définit ses attributs comme son contenu et sa date, son ID du nom d'TodoID est un integer qui sera auto incrémentée à chaque nouveau Todo.

Le dernier attribut est lid qui est une référence à l'id InterId de l'interaction.

Donc quand on créer une interaction a un contact , Le Cid de l'interaction prend la valeur du mail du contact et lorsque l'on crée un ou plusieurs Todo , les lid de ces todo prendront l'id InterID de l'interaction à laquelle ils appartiennent.

Voici un diagramme des tables :



Conclusion :

L'application permet à l'utilisateur de créer des contacts et d'interagir avec celui, l'application est designée de sorte à ce que l'utilisateur ai un visuel constant sur la liste de contact ainsi que la méthode de recherche tout en naviguant entre les différentes fonctions qui lui sont à disposition en plus de pouvoir exporter un fichier JSON et d'afficher l'historique de l'application.

Sources :

Pour le Qcalendar :

<https://doc.qt.io/qt-6/qcalendarwidget.html>

<https://doc.qt.io/qt-6/qdate.html>

Pour les images :

<https://doc.qt.io/qt-6/qpixmap.html>

<https://openclassrooms.com/forum/sujet/qt-qpixmap-52183>

<https://doc.qt.io/qt-6/qimage.html>

Pour les QTableWidgetItem :

<https://doc.qt.io/qt-6/qtablewidgetitem.html>

<http://www.java2s.com/Code/Cpp/Qt/QTableWidgetandQTableWidgetItem.htm> (utilisation image dans item)

Pour le telechargement d'images :

<https://en.cppreference.com/w/cpp/filesystem>

<https://en.cppreference.com/w/cpp/numeric/random>