

Rendu final

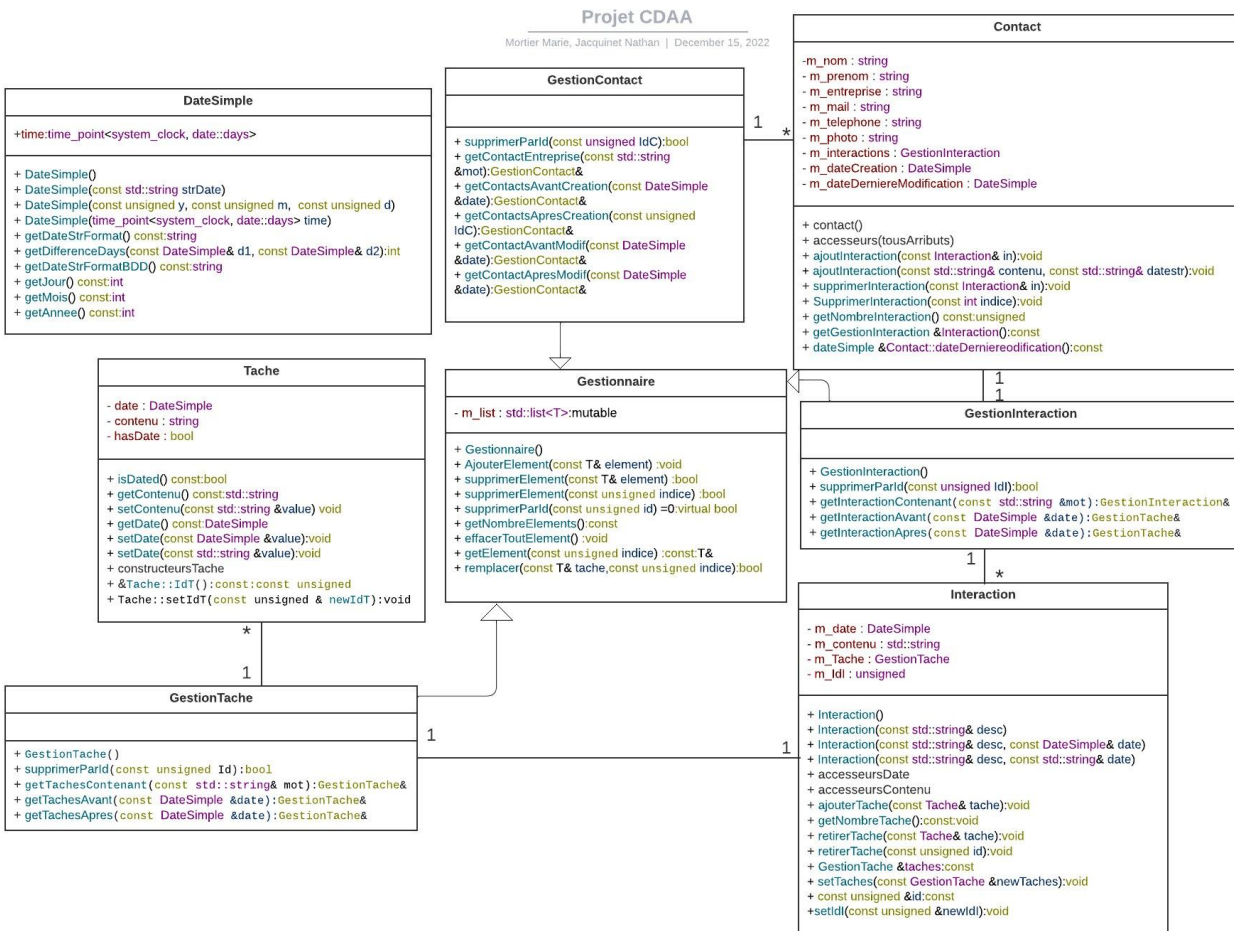
Introduction:

De gérer une connexion à une base de données, d'avoir une interface IHM pour faire la jointure entre les structures de stockage et les traitements internes.

Durant le premier rapport, nous avons conçu le cœur du programme, les classes métiers.

Nous avons fait une classe abstraite Gestionnaire, et chaque classe de gestion dérive de celle-ci: GestionContact, GestionTache, GestionInteraction.

Diagramme UML primitive:



Conception IHM:

Fenêtre principale:

Pour la conception de l'Interface Homme Machine, nous avons décidé d'afficher, sous forme de liste de Widget customisé (ContactWidget) un récapitulatif des informations liées aux contacts.
Ce Widget possède ce modèle:

The diagram shows a rectangular widget with a light gray background. On the left side, there is a square placeholder for a profile picture. To the right of the placeholder, the fields are arranged as follows:

- Prénom: prenom Nom : nom
- Téléphone: tel | Email: mail
- Entreprise: entreprise
- Date de création: dateCrea Dernière modification: dateModif

La fenêtre principale, Accueil, aura donc cette conception:

The diagram shows a window titled 'Accueil' with a menu bar containing 'Accueil', 'Affichage', 'Recherche', and 'Taper ici'. Below the menu bar, there is a section titled 'Rechercher par filtre :'. This section contains three search filters, each with a checkbox and a text input field:

- ☐ Nom [text input]
- ☐ Prenom [text input]
- ☐ Entreprise [text input]

Below these filters, there are two date selection sections:

- Date de création:
 - ☐ Après [31/12/1999] [calendar icon]
 - ☐ Avant [31/12/1999] [calendar icon]
- Date de modification:
 - ☐ Après [31/12/1999] [calendar icon]
 - ☐ Avant [31/12/1999] [calendar icon]

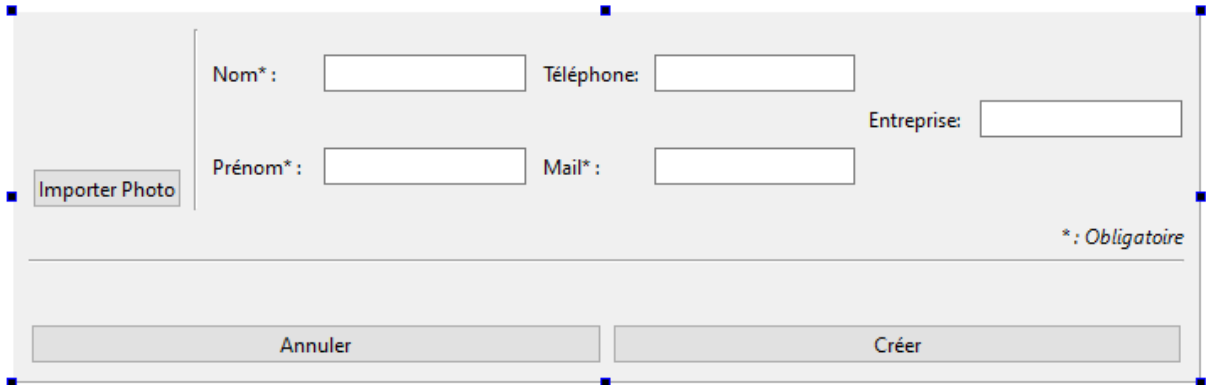
Below the filters and date sections is a large empty rectangular area for displaying the list of contacts. At the bottom of the window, there are two buttons: 'Supprimer' and 'Ajouter'.

Les filtres sont disposés en haut de la page, et chaque filtre est cumulable avec tous les autres.

En bas de la page, dans la barre de status sera affichée des informations comme le nombre total de contacts et le nombre de contacts actuellement visible.

Toujours en bas, 2 boutons pour créer un nouveau contact (Qui ouvrira une nouvelle fenêtre), et un autre pour supprimer le contact actuellement sélectionné dans la liste.

La fenêtre de création possède ce visuel:

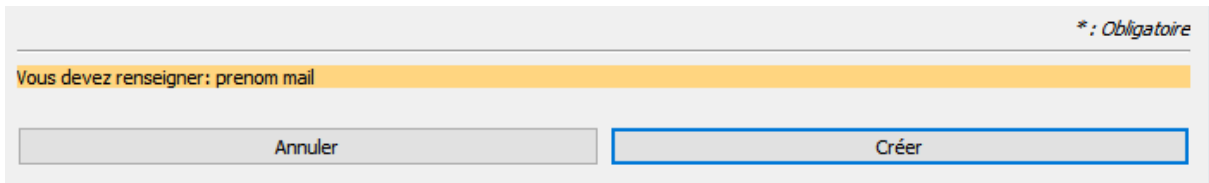


The screenshot shows a contact creation form with the following elements:

- A button labeled "Importer Photo" on the left.
- Input fields for "Nom*" (Name), "Téléphone:" (Phone), "Prénom*" (First Name), "Mail*" (Email), and "Entreprise:" (Company).
- A legend at the bottom right indicating "* : Obligatoire" (Mandatory).
- Two buttons at the bottom: "Annuler" (Cancel) and "Créer" (Create).

La fenêtre contrôle les champs qui ont été remplis:

Si un des champs obligatoire (noté *) n'est pas rempli, la création n'aura pas lieu, et un message visuel sera affiché, précisant en plus quel champ doit être rempli.



The screenshot shows the form with a validation error message:

- A yellow banner at the top reads "Vous devez renseigner: prenom mail" (You must provide: first name email).
- The "Créer" button is highlighted with a blue border.
- The "Annuler" button is visible on the left.
- The legend "* : Obligatoire" is still present at the top right.

Ouverture d'une fiche d'un contact:

Pour ouvrir la fiche d'un contact, il suffit de double cliquer sur un contact affiché dans la liste, ce qui présentera plus en détails le contact:

En haut de la page nous avons les attributs (nom, prénom, photo, tel etc...) du contact.

Puis nous avons la liste d'interaction associé à ce contact, qui fonctionne différemment de la liste de contact de la fenêtre précédente:

Pour afficher le contenu de l'interaction, et ses tâches associées, il suffit de sélectionner l'interaction dans la liste. Ensuite dans le contenu en bas de la fenêtre, dans un encadré de texte, y est affiché l'interaction, ainsi que chacune des tâches associées.

Nous pouvons ajouter ou supprimer une interaction à l'aide des boutons juste en dessous de la liste d'interaction.


Et nous pouvons filtrer les interactions du contact en précisant une date, un intervalle de date.

MORTIER Marie
JACQUINET Nathan

Enfin, nous avons la possibilité d'afficher chaque tâches qui sont présentes dans la liste en cliquant sur le bouton Toutes les tâches.

Voici un aperçu de la fenêtre de la fiche d'un contact:

Fiche contact: Mortier Marie



Mortier

marie.mortier@gmail.com

Marie

0645874920

Etudiant

Interaction:

Toutes les tâches

☐ Après 16/12/2022

☐ Avant 16/12/2022

Continuer SR, juste le rapport	15/12/2022	2
Noel	24/12/2022	1

Quitter

Ajouter

Supprimer

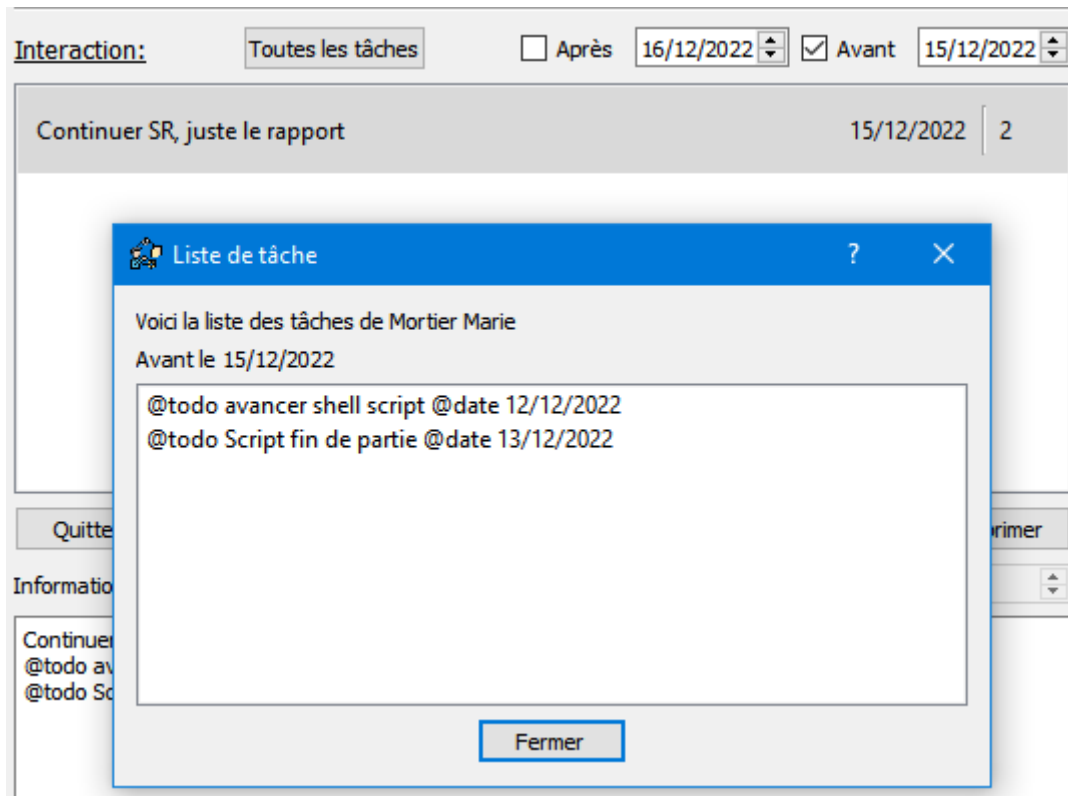
Informations sur l'interaction:

15/12/2022

Continuer SR, juste le rapport
@todo avancer shell script @date 12/12/2022
@todo Script fin de partie @date 13/12/2022

Double cliquez sur le contenu pour modifier le contenu de l'interaction

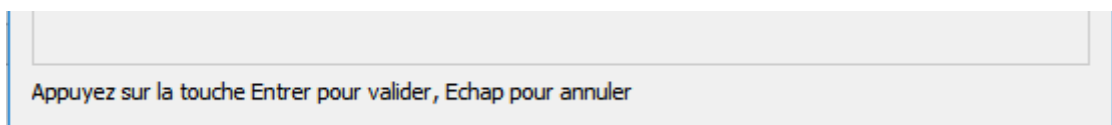
Affichage de toutes les tâches après un filtrage: (Ici est affiché toutes les tâches après le 15/12/2022)



Modifier un contact, et ajouter du contenu aux interactions:

Pour modifier le contact, il suffit de cliquer sur un attribut du contact, par exemple son nom. Cela nous fait entrer en mode édition, cela nous bloque ainsi tout le reste du contenu jusqu'à ce qu'on sorte du mode édition. Pour en sortir, il faut soit valider la modification (à l'aide de la touche Entrer) soit annuler la modification. (Avec la touche Echap)

Toutes ces informations du mode édition sont affichées en bas dans la barre de status:



MORTIER Marie
JACQUINET Nathan

Visuel du mode édition:

The screenshot shows a window titled 'Fiche contact: Mortier Marie'. It contains a profile picture of an owl, a text input field with 'Mortier', an email address 'marie.mortier@gmail.com', a name 'Marie', a phone number '0645874920', and a role 'Etudiant'. Below this is an 'Interaction:' section with a dropdown menu set to 'Toutes les tâches', a checkbox for 'Après' (unchecked), a date picker for '16/12/2022', a checkbox for 'Avant' (checked), and another date picker for '15/12/2022'. A table below shows one interaction: 'Continuer SR, juste le rapport' with a date of '15/12/2022' and a count of '2'.

Modifier une interaction, et rajouter des tâches:

Pour modifier une interaction, même système:

Il suffit de cliquer sur le champ de texte contenant les infos de l'interaction.

Nous entrons de nouveau en mode édition, nous pouvons ajouter, supprimer des tâches, modifier les dates etc...

Pour sauvegarder les modifications, il suffit de faire la combinaison de touches: CTRL-s.

Affichage du mode édition de l'interaction:

The screenshot shows a window for editing an interaction. It has buttons for 'Quitter', 'Ajouter', and 'Supprimer'. Below these is a section 'Informations sur l'interaction:' with a date picker set to '15/12/2022'. A text area below contains the following text: 'Continuer SR, juste le rapport', '@todo avancer shell script @date 12/12/2022', and '@todo Script fin de partie @date 13/12/2022'.

Fonctionnalité supplémentaire:

Charger une nouvelle base de donnée: Fenêtre Accueil, Onglet Accueil -> Charger -> depuis BDD
La base de données doit être conforme au schéma décrit dans la partie BDD.

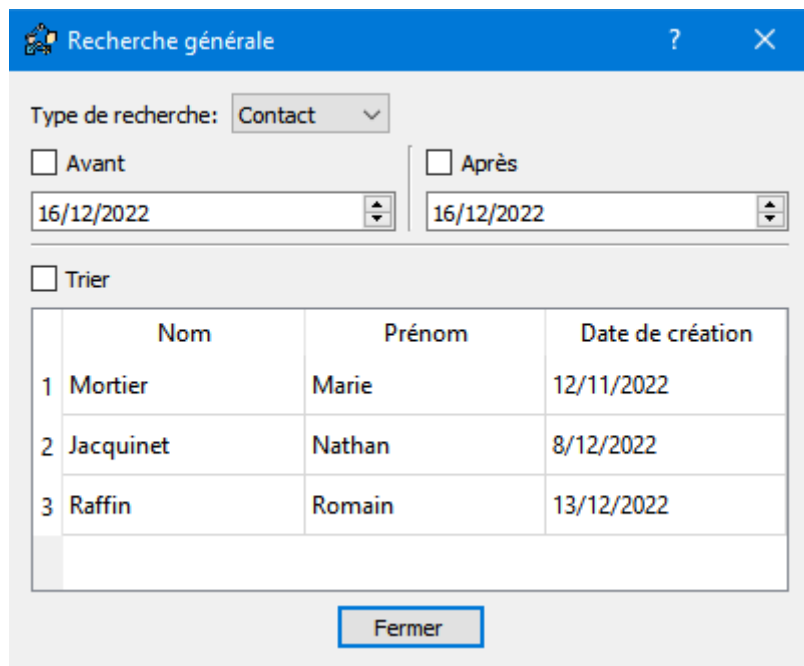
Réactualiser les données: Fenêtre Accueil, Onglet Affichage -> Actualiser
Permet de forcer l'actualisation de l'affichage des données.

Exporter les données au format JSON: Fenêtre principale -> Accueil -> Exporter -> JSON

MORTIER Marie
JACQUINET Nathan

Exporte les données de l'application au format JSON, en supprimant tous les identifiants des instances (qui sont associé à la BDD de l'utilisateur.)
Cela permet donc une portabilité maximale.

Faire des recherches sur le jeu de donnée entier: Fenêtre Accueil -> Recherche -> Générale
Une nouvelle fenêtre s'ouvre, permettant de filtrer par date, intervalle de date tout type de donnée (Contact, interaction, tâche)
De les trier par ordre alphabétique, anti-alphabétique, par date croissant, décroissant.



	Nom	Prénom	Date de création
1	Mortier	Marie	12/11/2022
2	Jacquinet	Nathan	8/12/2022
3	Raffin	Romain	13/12/2022

Choix de lien entre l'interface IHM et le coeur du programme:

Pour tout ce qui est de l'ajout, de la suppression etc des instances de classe métier, ce sont uniquement elles qui peuvent le faire.

Lors de la lecture de documentation, et de test en interne, les

Cela pose deux principes qui ont été respecté dans le projet:

Nous autorisons l'utilisation de pointeur vers les instances des classes métier, pour les modifier ou lire les données, mais en aucun cas les classes "utilisatrices", celles qui possèdent ces pointeurs, ne sont pas propriétaires des instances et elles se doivent donc de ne jamais supprimer une instance.

En d'autre terme, les instances des Contact, interactions ainsi que les tâches ne sont présentes dans la RAM qu'une seule fois.

Des fonctions des tri de liste, qui sont dans les classes GestionTache par exemple, sont particulièrement lourdes et ont pu être évitées.

Car elles suppriment toutes les instances qui ne correspondent pas au critère du filtre, or il faut donc au préalable avoir fait une copie entière de l'instance de notre gestionnaire, avant d'appliquer les filtres.

Cependant les fonctions sont disponibles.

Comment avons-nous fait les filtres alors ?

Elles sont faites en interne dans des fonctions dédiés qui agissent les les items des listes (celle présente dans la page d'Accueil, mais aussi celle dans la fiche de contact et la fenêtre de recherche générale)

Ces fonctions se contentent juste de rendre invisible l'item, et les faire réapparaître dès lors qu'il satisfait le filtre!

Cela nous permet d'avoir une certaine optimisation, car nous ne faisons aucune copie d'instance, et aucune fuite de mémoire n'existe tant que ce principe de conception est respecté.

Nous verrons des exemples d'utilisation de ces pointeurs après.

Jointure entre données de l'application, et une base de donnée SQLite:

Comme vu auparavant, l'IHM utilise des pointeurs directement vers les instances des classes métier.

Le gestionnaire de base de données (MainSqlManager) garde ce principe: toutes ses fonctions membres utilisent des pointeurs vers des instances de classe.

Elle n'ajoute pas, ni ne supprime d'instance dans les gestionnaires.

Elle s'occupe uniquement de la BDD.

Diagramme classe mainsqlmanager:

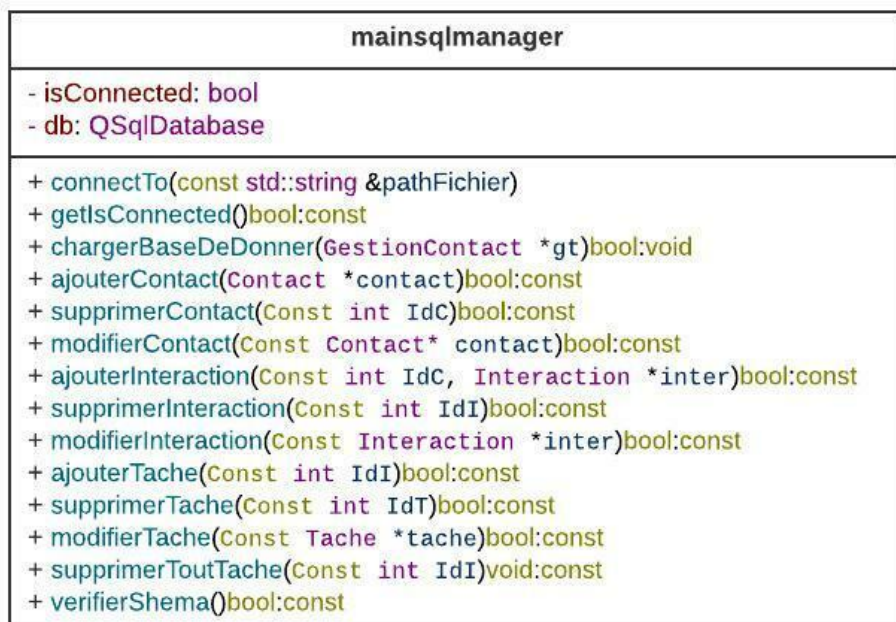
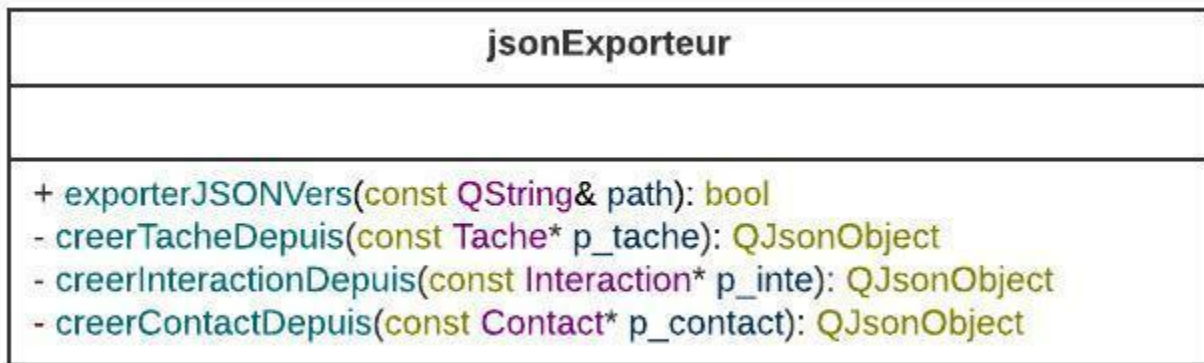
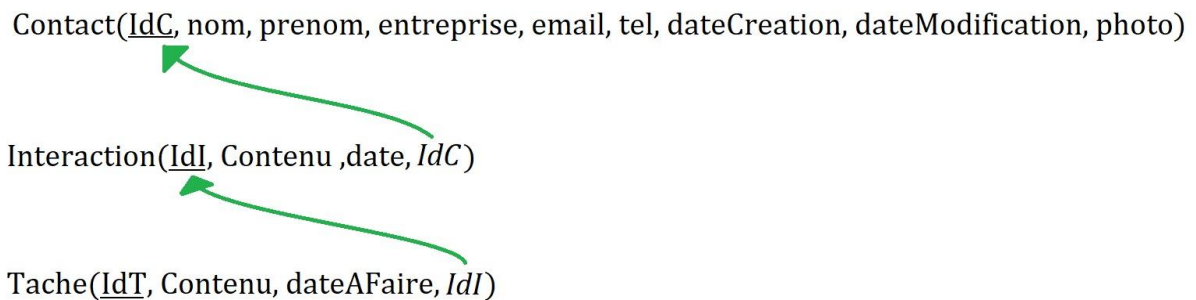


Diagramme classe JsonExporteur:



Le programme utilise une base de donnée, composée de 3 tables, ayant des schéma, et ses liens:



Des contraintes ont été mis sur les tables Interaction ainsi que Tâche:

ON DELETE CASCADE des interactions ayant pour valeur de clef étrangère un contact qui a été supprimé de la table Contact.

De même sur la Table avec les Interactions

Le fichier générant ces tables ainsi que quelques entrée dans les tables est *createTable.sql*

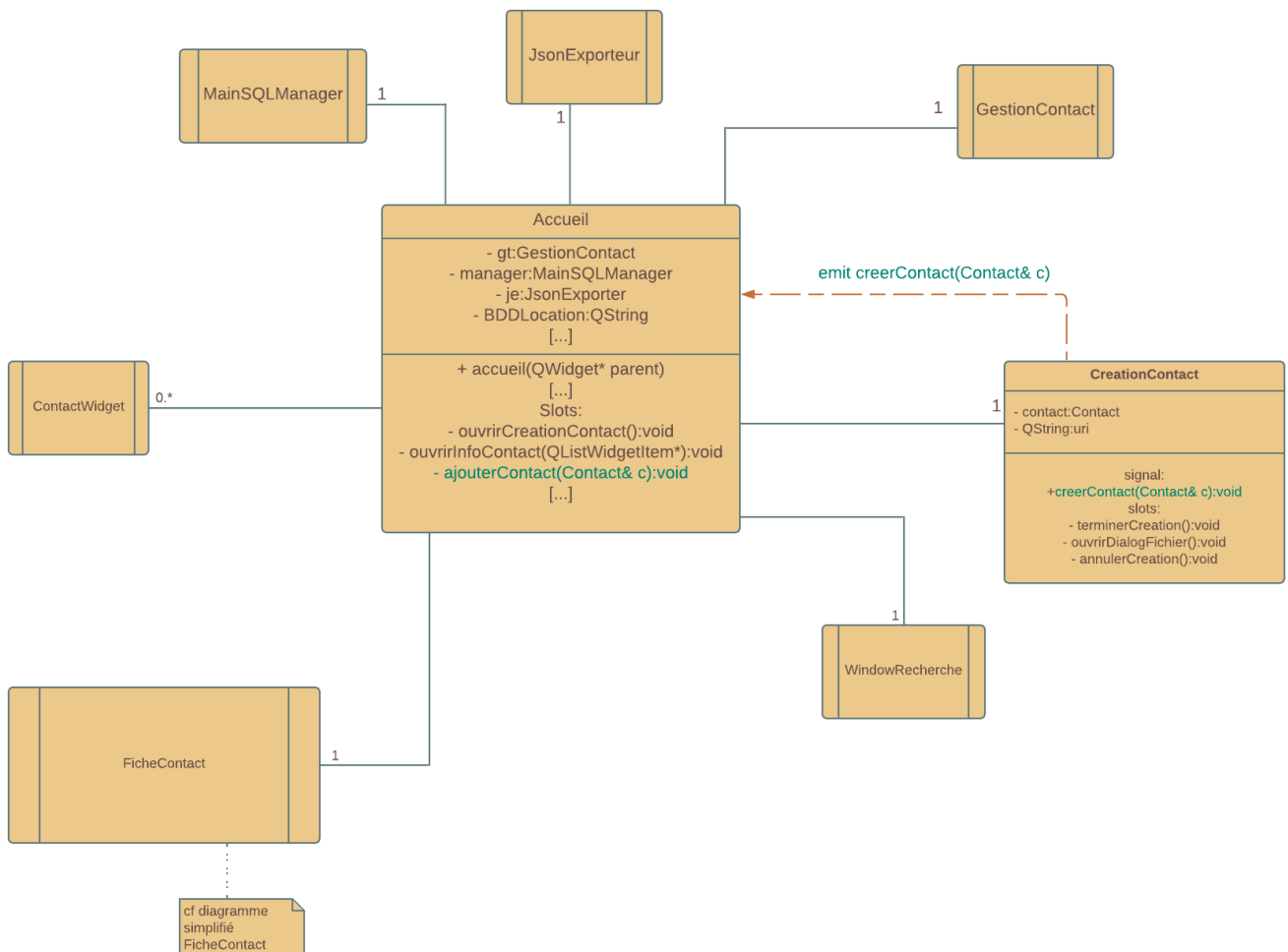
Explication relations entre les classes de l'IHM et les autres classes:

Introduction:

Le programme utilise exclusivement une QMainWindow, et les autres fenêtres sont des dérivées de QDialog.

La raison est qu'une QDialog peut bloquer la fenêtre parent.
Cela permet d'éviter d'ouvrir de multiple fenêtre à en perdre la vue.

Classe accueil:



La classe Accueil est la classe permettant l'affichage de la fenêtre principale. Elle est la jointure entre l'IHM, les gestionnaires de stockage (BDD) et le coeur du programme (classes métier)

Elle permet entre autre d'ouvrir quelques petite fenêtre (comme WindowRecherche, ou CreationContact)

MORTIER Marie
JACQUINET Nathan

Elle reçoit un signal envoyé par l'instance CreationContact et ajoute ensuite le nouveau contact dans le GestionContact, ainsi que dans la base de données.

Il y a beaucoup de slots interne à la classe Accueil, ceux reliant les checkbox, les boutons, les lineEdit ainsi que les actions dans la barre d'action.

Si le comportement de ces slots vous intéresse, ils seront résumés dans la documentation.

Classe CreationContact:

Cette classe, tout comme le diagramme de classe association orienté sur Accueil le montre, elle émet un signal avec en paramètre le contact fraîchement créé.

Elle traite les signaux de ses composants graphiques en interne, comme bien souvent les classe IHM de ce projet.

Classe FicheContact:

Cette classe joue un rôle important dans l'application:

Elle n'a pas autant de rôle que la classe principale, mais elle s'occupe exclusivement et à elle seule de la modifications des données d'un Contact (attribut, interactions, tâche etc..)

Elle traite uniquement les signaux en interne, mais possède des algorithmes et des fonctions variées pour extraire des données, créer de nouvelles données, en supprimer etc...

Je vous invite encore une fois à voir la documentation lié à cette classe importante.

Elle réceptionne des signaux émis par des instance de classe QLabel et QTextEdit customisé:
En effet ces redéfinitions émettent maintenant un signal lorsqu'on double clique sur eux.

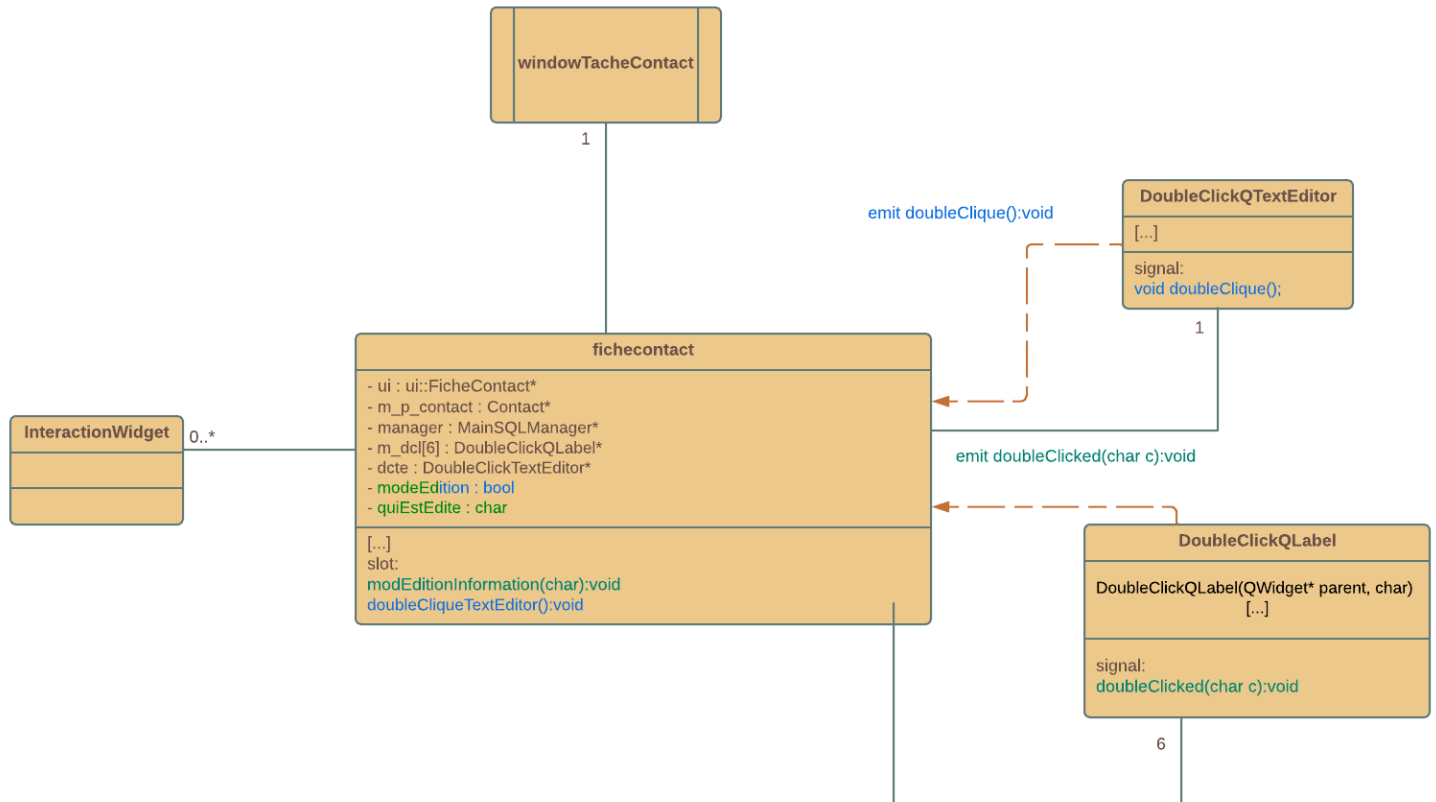
Suivant le char du signal réceptionné, la classe adapte en fonction de qui a émis ce signal: le char réceptionné est comme un identifiant du label qui l'a émis.

Extrait de code source:

- * **Explications:**
- * **Nous avons 6 pointeurs de notre QLabel custom**
- * **0: nom, 1: prenom, 2: tel, 3: mail, 4: entreprise, 5: photo**
- * **et en parallèle nous avons 5 QLineEdit**
- * **0: nom, 1: prenom, 2: tel, 3: mail, 4: entreprise**
- * **Celui de la photo sera géré par la QFileDialog (pour aller chercher une nouvelle photo)**
- *
- * **Chaque QLineEdit est dans un premier temps caché, il sera visible uniquement quand le label aura**
- * **reçu un double click, dans lequel cas on entrera en mode édition de cet attribut**
- * **du contact**

Je vous invite de nouveau à lire la documentation ainsi que le code source pour découvrir les algorithmes.

Voici un schéma simplifié des interactions qu'elle peut avoir avec les autres classes:



Pour toutes les autres classes qui n'ont pas eu d'explication, Doxygen génère ses liaisons avec les autres classes, bien que existante, elle n'en reste pas moins des classes qui pourraient être autonomes.

Quelques informations supplémentaires:

Pour les photos des Contacts, lors de la création (ou de la modification) de la photo, l'utilisateur doit ouvrir à l'aide d'un `QFileDialog` le fichier correspondant à la photo, s'il est conforme, une copie de cette photo est alors créée dans le dossier `images/N°IdContact.extension`

Ainsi, pour la mutation du logiciel vers un autre système, il suffira de copier la base de données, ainsi que le dossier photo.