# Conception et développement avancé d'application (CDAA)
# L3 Informatique
# Sujet de projet

R. Raffin

UFR des sciences et techniques
romain.raffin[at]u-bourgogne.fr

2022-23

## Organisation

À faire en **binôme**, 2 jalons pendant les TP + la revue lors du dernier TP. À rendre dans la semaine 50 du **12 au 15 décembre**, sur l'espace Plubel.

## Sujet

On veut développer une application QT qui permette de dialoguer avec une base de données SQL, afin de gérer une base de données de contacts et des interactions (comme un CRM [1]). On doit pouvoir :

1. **créer la fiche** d'un contact (nom, prénom, entreprise, mail, téléphone, photo, date de création),
2. **modifier un contact**, supprimer une fiche. Attention les opérations doivent être horodatées automatiquement : on doit être capable de savoir quand une fiche a été modifiée, ou de la date de la dernière suppression (pas de notion de corbeille, les fiches sont réellement supprimées),
3. **ajouter une interaction** : un rendez-vous, un commentaire... sous forme d'un champ texte mais horodaté. On doit donc pouvoir lister l'ensemble des interactions d'un contact, avec les dates de modification triées,
4. **« tagger » des mots** pour pouvoir les indexer dans la note liée à une interaction. Par exemple :

   ```
   rdv aujourd'hui par tél., RAS.
   @todo Rappeler @date 16/10/2021
   @todo confirmer commande n°xyz
   ```

   Cela permet de définir une date importante et un ajout à la liste des choses à faire (*todo* de ce contact). La formulation commence par `@todo`, l'action à effectuer (sur une seule ligne). Elle peut être suivi (sur la même ligne) d'un tag `@date` suivant d'une chaîne au format `jj/mm/aaaa`. Si le tag `@date` n'est pas présent, le `@todo` est à faire pour la date courante (de remplissage de cette interaction). Les `@todo` sans date (par défaut) sont donc les plus urgents.
5. **stocker** l'ensemble des éléments dans une base de données SQLite,
6. pouvoir **chercher un contact** par son nom ou son entreprise, dans un intervalle de dates, par sélection dans une liste triée alphabétiquement ou par date de création,
7. effectuer et afficher des **requêtes** : nombre de contacts, interactions entre 2 dates, collecter pour un contact et un intervalle de dates (1 semaine par exemple) la liste des @todo ou des dates @date, idem pour l'ensemble des contacts,
8. proposer un **export en JSON** des données pour faciliter l'interopérabilité.

## Erreurs à ne pas commettre, contraintes

Attention à ne pas lier les structures de données du CRM avec des types de QT pour faciliter la généricité de vos algorithmes. On peut imaginer que ces traitements puissent être compilés dans une bibliothèque (type `.so` ou `.dll`), sans appui sur QT. On utilisera donc les containers C++ (`std::list`, `vector`, `deque`, `map`...) dont on transformera les données pour l'interaction avec QT.

1. `https://fr.wikipedia.org/wiki/Gestion_de_la_relation_client`

Attention à la gestion des erreurs, à la manipulation de vos structures de données (ajout/modification/suppression de contact, interaction, contenu...) mais également lors du passage de l'IHM ou de la base de données à vos structures (dates bien construites, cohérentes, URI des photos n'étant pas des images, de format PNG ou JPG). Ces vérifications peuvent se faire dans chaque module en s'appuyant sur les fonctions intrinsèques (ouverture d'un calendrier pour saisir une date dans QT permet d'être sûr de la saisie - pas forcément de la cohérence si l'utilisateur sélectionne une `@date` pour le 01/01/1970, les constructeurs de données peuvent renvoyer une erreur). Utilisez massivement les exceptions.

Les sources cpp/h doivent être documentées avec **Doxygen** et les commentaires du code doivent permettre la documentation du projet : classes, algorithme, fonction. Le diagramme des classes sera généré par Doxygen.

Le rendu s'effectuera sous forme d'une archive **zip** contenant seulement les sources `c++/h`, le `.pro` de QT, le fichier de configuration Doxygen, un fichier *SQLite* initial, les données et ressources nécessaires à l'application. Le modèle de données SQL sera fourni dans un fichier PDF. Joignez un fichier PDF ou markdown pour les besoins de librairies et la description de l'installation de votre application.

**L'archive sera remise sur Plubel, avant la date de fin du projet !**

Pour faciliter la gestion des dates, et parce que la gestion des dates est prise en compte à partir de c++20, ce qu'il n'y a pas forcément sur toutes les machines de TP, on utilisera le *header* « *date.h* » :

`https://howardhinnant.github.io/date/date.html`

Une alternative est l'usage de *ctime*, avec une manipulation plus forte de pointeurs.

*En bonus (si tous les points précédents sont respectés, mais qui peut permettre de compenser des manques ou des défauts) : application Android (déposer l'apk) et/ou MsWindows, internationalisation de l'application, décorateurs de table QT pour l'affichage, installateur...*