République Tunisienne
Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique
Université de Tunis El Manar
École Nationale d'Ingénieurs de Tunis

Département Génie Électrique

Mini Projet C++

Projet 2 :

Carnet de contacts

Réalisé par :
Yassin OMRI

Classe :
2AGE1

Encadré par :
Mme. Amira KALLEL

Année universitaire 2023/2024

Table des matières

Table des figures

Introduction générale

À l'ère de la connectivité omniprésente, la gestion des contacts est devenue une nécessité tant dans la sphère personnelle que professionnelle. La capacité à organiser efficacement les informations de contact facilite la communication et favorise la collaboration. Dans ce contexte, le développement d'une application de carnet de contacts offre une solution moderne et pratique pour centraliser et gérer les coordonnées de diverses personnes.

Ce rapport présente le projet 2 intitulé "Carnet de contacts", réalisé dans le cadre du cours de C++ de l'École Nationale d'Ingénieurs de Tunis. L'objectif de ce projet est de concevoir et implémenter une application graphique permettant de gérer efficacement un carnet de contacts. Cette application offrira des fonctionnalités telles que l'ajout, la modification, la suppression et la recherche de contacts, tout en garantissant une interface utilisateur conviviale et intuitive.

Dans les sections suivantes, nous détaillerons les différentes étapes de développement de l'application, en mettant l'accent sur les choix de conception, les fonctionnalités implémentées et les défis rencontrés. Enfin, nous discuterons des perspectives d'amélioration et des potentielles extensions de ce projet.

Ce projet vise à développer une application graphique de gestion de carnet de contacts, offrant à l'utilisateur une interface intuitive pour gérer efficacement ses contacts. Les fonctionnalités principales incluent l'ajout, la modification, la suppression et l'affichage des contacts, ainsi que des fonctionnalités avancées telles que la recherche et le tri. De plus, l'application offrira la possibilité de visualiser des statistiques détaillées sur les appels.

Les données des contacts seront stockées de manière sécurisée dans une base de données, assurant une gestion centralisée et une manipulation efficace des informations, même pour un grand nombre de contacts.

Ce rapport détaillera le processus de développement de l'application, en mettant en lumière les fonctionnalités implémentées, les choix technologiques et les défis rencontrés. Des captures d'écran illustreront le fonctionnement de l'application, tandis que des explications détaillées fourniront un aperçu du développement.

Chapitre 1

Cahier des charges du projet

1.1 Introduction

Dans ce chapitre, nous élaborons le cahier des charges de notre projet de gestion de carnet de contacts, précisant les objectifs, les fonctionnalités et les exigences techniques.

1.2 Objectif

L'objectif principal est de concevoir une application permettant de gérer efficacement un carnet de contacts, en offrant des fonctionnalités essentielles telles que l'ajout, la modification, la suppression et l'affichage des contacts. De plus, l'application doit permettre des fonctionnalités avancées telles que la recherche, le tri et la visualisation des statistiques.

1.3 Besoins techniques

1.3.1 Besoins fonctionnels

Les besoins fonctionnels incluent la gestion des contacts, la gestion des appels et la visualisation des statistiques. Cela garantira une expérience utilisateur complète pour la gestion des contacts et des appels, ainsi qu'une analyse approfondie des habitudes d'appels.

1.3.2 Besoins non fonctionnels

Les besoins non fonctionnels incluent la performance, la fiabilité, la sécurité, la facilité d'utilisation et l'extensibilité de l'application, assurant une expérience utilisateur fluide et sécurisée, ainsi qu'une adaptabilité aux besoins évolutifs.

1.4 Outils et technologies

1.4.1 Qt Creator

Qt Creator sera utilisé comme environnement de développement pour sa richesse en outils dédiés au développement d'applications multiplateformes, notamment son éditeur de code sophistiqué, son concepteur d'interfaces graphiques et son débogueur visuel.

FIGURE 1.1 – Logo du Qt Creator

1.4.2 SQLite

SQLite sera utilisé comme système de gestion de base de données pour sa légèreté, sa rapidité et sa portabilité, offrant une solution efficace pour le stockage et l'accès aux données de l'application.

1.4.3 DB Browser

DB Browser sera utilisé pour la gestion de la base de données SQLite, offrant une interface graphique conviviale pour la création, la modification et la navigation des données, ainsi que la sauvegarde et la récupération des bases de données.

FIGURE 1.2 – Logo du DB Browser

1.4.4 Langage C++

Le choix du langage de programmation C++ s'est imposé pour le développement de notre application de carnet de contacts en raison de sa robustesse, de sa flexibilité et de sa performance. En tant que langage orienté objet, C++ permet une modélisation

efficace des données à travers l'utilisation de classes et de structures, favorisant ainsi la maintenabilité et la réutilisabilité du code. De plus, sa gestion précise de la mémoire offre un contrôle accru sur les ressources système, ce qui est essentiel pour garantir des performances optimales de l'application.

1.5 Modélisation des Données et Fonctionnalités

1.5.1 Introduction

Dans cette section, nous explorerons la modélisation des données de notre application de carnet de contacts ainsi que les principales fonctionnalités qu'elle offrira.

1.5.2 Modélisation des Données

La modélisation des données repose sur la conception de la structure de la base de données utilisée par l'application. Pour notre projet, nous avons identifié trois principales entités :

— **Carnet** : Cette entité comprendra les informations telles que le nom, le numéro de téléphone, l'adresse et l'adresse e-mail des contacts.
— **Appel** : Cette entité enregistrera les détails des appels effectués, y compris le nom de l'appelant, l'heure de début et de fin de l'appel, ainsi que sa durée.
— **Login** : Cette entité gérera les informations d'authentification des utilisateurs, y compris leur nom d'utilisateur, leur mot de passe et leur rôle (utilisateur ou administrateur).

La figure suivante illustre la structure de ces entités dans la base de données SQLite :

Name	Type	Schema
Tables (4)		
appel		CREATE TABLE "appel" ("idappel" INTEGER
idappel	INTEGER	"idappel" INTEGER NOT NULL UNIQUE
debut	TEXT	"debut" TEXT
fin	TEXT	"fin" TEXT
dure	TEXT	"dure" TEXT
nom	TEXT	"nom" TEXT
tel	INTEGER	"tel" INTEGER
id	INTEGER	"id" INTEGER
carnet		CREATE TABLE "carnet" ("idcontact" INTEC
idcontact	INTEGER	"idcontact" INTEGER NOT NULL UNIQUE
nom	TEXT	"nom" TEXT
adresse	TEXT	"adresse" TEXT
tel	INTEGER	"tel" INTEGER NOT NULL
email	INTEGER	"email" INTEGER
login		CREATE TABLE "login" ("idlogin" INTEGER I
idlogin	INTEGER	"idlogin" INTEGER NOT NULL
User_name	TEXT	"User_name" TEXT
PSW	TEXT	"PSW" TEXT
Role	TEXT	"Role" TEXT

FIGURE 1.3 – Structure de la base de données

1.5.3 Fonctionnalités du Projet

Les fonctionnalités principales de notre application de carnet de contacts seront les suivantes :

- **Gestion des Contacts** : Ajout, modification, suppression et affichage des contacts.
- **Gestion des Appels** : Réception et enregistrement des appels, y compris la mise à jour des informations relatives aux appels.
- **Authentification** : Système de connexion sécurisé avec des rôles d'utilisateur et d'administrateur.

Ces fonctionnalités garantiront une expérience utilisateur complète et sécurisée pour la gestion des contacts et des appels.

1.6 Conclusion

Cette section a fourni un aperçu de la modélisation des données de notre application de carnet de contacts, ainsi que des principales fonctionnalités qu'elle offrira. Ces éléments sont essentiels pour définir le cadre du développement de notre projet et assurer sa fonctionnalité et sa sécurité.

Chapitre 2

Réalisation du Projet

2.1 Introduction

Ce chapitre détaille la mise en œuvre de notre application de carnet de contacts, en mettant en évidence chaque aspect de son interface utilisateur ainsi que la conception et l'implémentation de ses fonctionnalités. Nous explorerons également les choix technologiques effectués et les différentes étapes du processus de développement du système.

2.2 Interfaces Utilisateurs

2.2.1 Interface de Connexion

L'interface de connexion offre aux utilisateurs la possibilité de se connecter à l'application en saisissant leur nom d'utilisateur et leur mot de passe, ainsi qu'en sélectionnant leur rôle (administrateur ou utilisateur). Une fois les informations validées, l'application vérifie les identifiants dans la base de données SQLite via Qt Creator en C++. En cas de succès, l'utilisateur est redirigé vers l'interface correspondant à son rôle, sinon un message d'erreur s'affiche pour l'inviter à réessayer.

FIGURE 2.1 – Interface de Connexion

2.2.2 Interface Administrateur

L'interface administrateur accueille l'utilisateur avec une image thématique et lui offre la possibilité d'interagir avec les contacts. L'administrateur peut ajouter, supprimer, afficher, initialiser et mettre à jour les données des contacts à travers une interface conviviale.

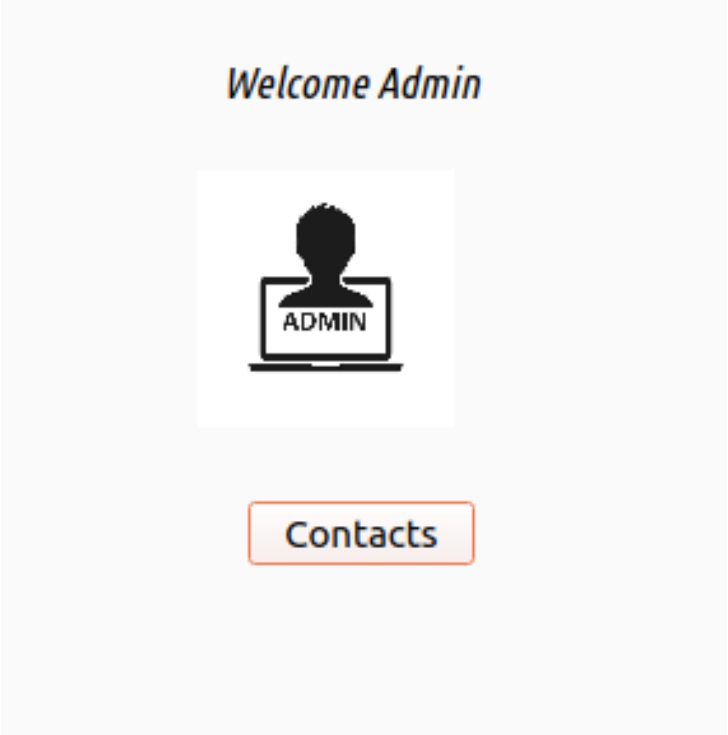

FIGURE 2.2 – Interface Administrateur

2.2.3 Interface Utilisateur

L'interface utilisateur propose quatre onglets : "Appel", "Recherche", "Stats", et "Histogramme". Dans l'onglet "Appel", l'utilisateur peut sélectionner un contact pour démarrer un appel et enregistrer les détails de l'appel. Dans l'onglet "Recherche", il peut rechercher des contacts par nom ou numéro. Dans l'onglet "Stats", il peut générer des statistiques sur les appels. Enfin, dans l'onglet "Histogramme", l'utilisateur peut voir des graphiques qui traduisent les statistiques, mais sous une forme visuelle

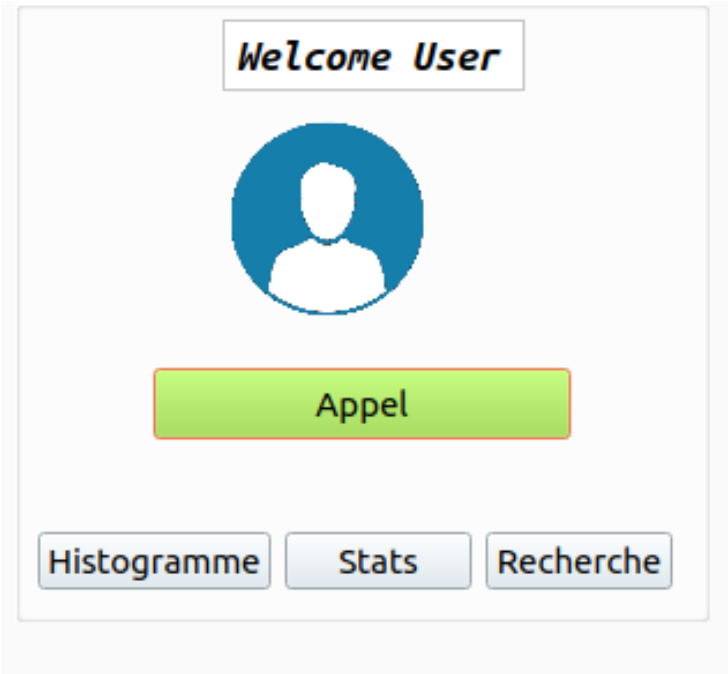

FIGURE 2.3 – Interface Utilisateur

2.3 Description des Classes

2.3.1 Classe de Connexion à la Base de Données "dbconnexion"

La classe de connexion "dbconnexion" gère l'accès à la base de données SQLite. Elle fournit des méthodes pour établir et fermer la connexion, masquant ainsi les détails de l'implémentation de la base de données. L'utilisation de cette classe améliore la modularité et la maintenabilité de l'application.

2.3.2 Classe Carnet

La classe des contacts "carnet" gère les opérations liées aux contacts dans la base de données. Elle offre des méthodes pour récupérer, définir, ajouter, supprimer et mettre à jour les contacts, assurant ainsi la cohérence des données. De plus, elle propose une méthode pour charger tous les contacts depuis la base de données.

Ci-dessous est présentée le contenu du fichier "carnet.h" :

```
6  class carnet
7  {
8  public:
9      carnet();
10     QString get_idcontact() ;
11     QString get_nom();
12     QString get_tel() ;
13     QString get_adresse() ;
14     QString get_email() ;
15
16     void set_idcontact(QString);
17     void set_nom(QString);
18     void set_tel(QString);
19     void set_adresse(QString);
20     void set_email(QString);
21
22     void Ajouter ();
23     void Supprimer();
24     void Modifier();
25     QSqlQueryModel *load () ;
26
27 private :
28
29     QString idcontact;
30     QString nom;
31     QString tel ;
32     QString adresse;
33     QString email;
```

FIGURE 2.4 – Déclaration de classe Carnet

2.3.3 Classe contactDialog

La classe ‘contactDialog‘ a été développée pour gérer les interactions de l’utilisateur avec les contacts dans notre application. Elle fournit une interface conviviale permettant d’ajouter, de modifier, de supprimer et de charger les données des contacts à partir de la base de données SQLite.

Elle encapsule les fonctionnalités nécessaires pour effectuer ces opérations, notamment la connexion à la base de données, la récupération des données saisies par l’utilisateur, et l’exécution des requêtes SQL correspondantes.

L’interface graphique de la classe ‘contactDialog‘ est présentée ci-dessous :

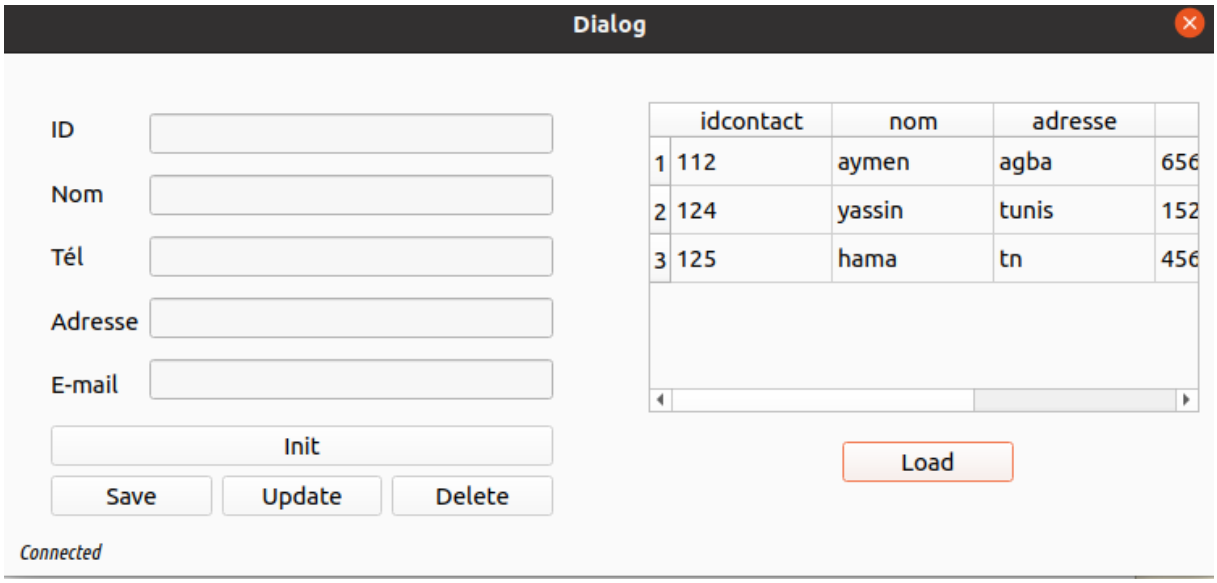

FIGURE 2.5 – Contact Dialog

2.3.4 Classe AppelDialog

La classe ‘AppelDialog‘ offre une interface conviviale pour gérer les fonctionnalités liées aux appels au sein de l’application. Elle facilite l’interaction de l’utilisateur avec les appels en fournissant des méthodes pour répondre ou refuser un appel, ainsi que pour enregistrer les détails de l’appel dans la table "Appel".

Lors de son initialisation, cette classe récupère les noms des contacts à partir de la base de données et les affiche dans une ComboBox, permettant à l’utilisateur de sélectionner l’appelant. Un chronomètre est également initialisé pour mesurer la durée de l’appel en cours.

L’interface graphique de la classe ‘AppelDialog‘ est présentée ci-dessous :

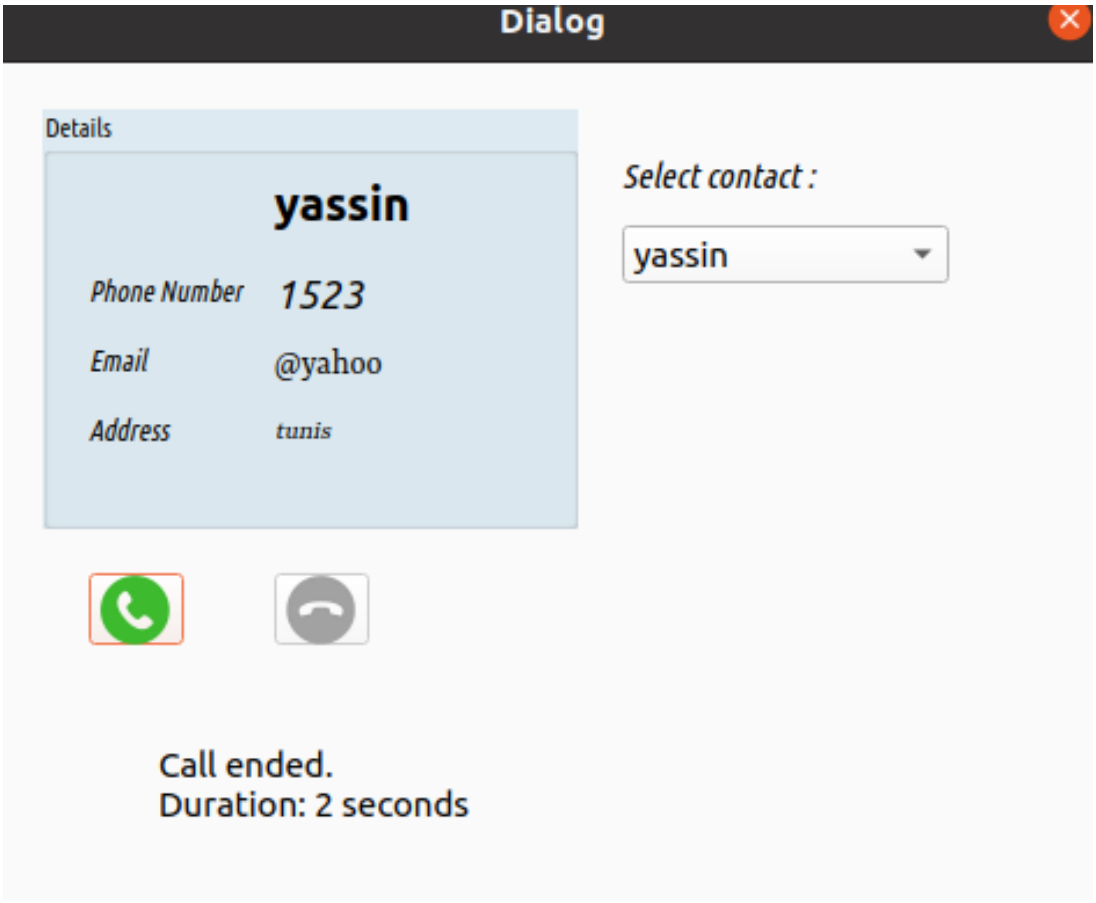

FIGURE 2.6 – Appel Dialog

2.3.5 Classe rechDialog

La classe ‘rechDialog‘ a été développée pour gérer les opérations de recherche au sein de notre application. Elle permet à l'utilisateur d'effectuer des recherches en fonction des critères sélectionnés (recherche par numéro, par nom), en construisant dynamiquement une requête SQL pour filtrer les résultats.

Elle encapsule les fonctionnalités nécessaires à la mise en œuvre de ces opérations, notamment la récupération des critères de recherche saisis par l'utilisateur et l'exécution des requêtes SQL correspondantes.

En résumé, la classe ‘rechDialog‘ joue un rôle essentiel dans notre application en permettant à l'utilisateur d'explorer et de récupérer facilement les données souhaitées à partir de la base de données SQLite.

Ci-dessous est présentée l'interface graphique de la classe "rechDialog" :

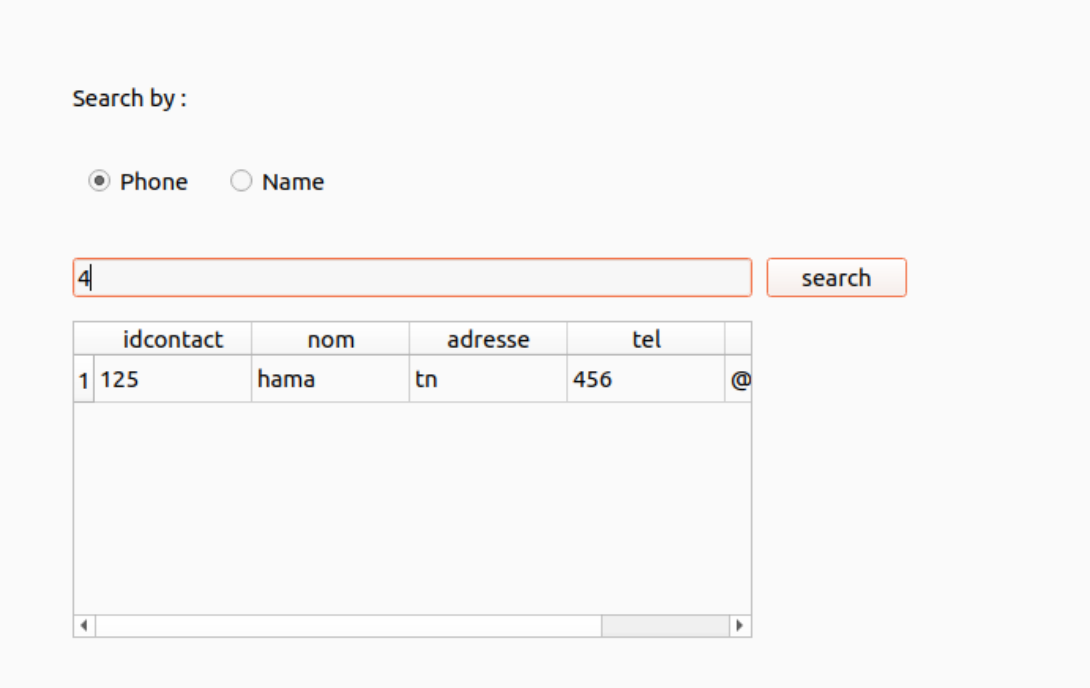

FIGURE 2.7 – Recherche selon le nom

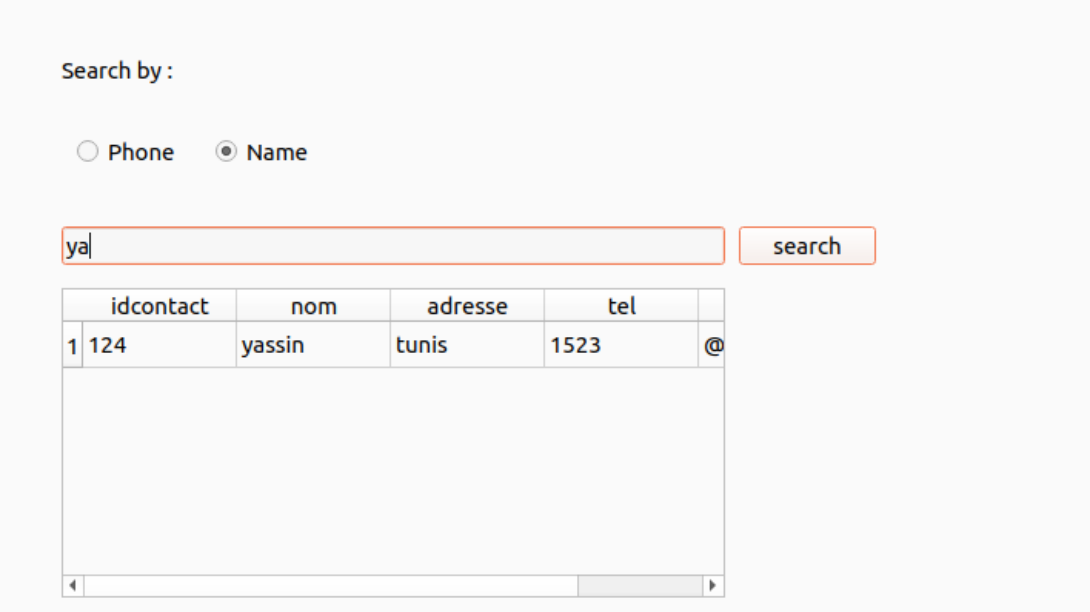

FIGURE 2.8 – Recherche selon le numéro de téléphone

2.3.6 Classe Histogram

La classe 'Histogram' joue un rôle essentiel dans l'application en permettant la représentation visuelle des données sous forme d'histogramme. Sa fonction principale est de fournir une interface graphique pour afficher efficacement les données, ce qui facilite la compréhension et l'analyse des informations.

Elle offre un ensemble de fonctionnalités permettant de personnaliser l'apparence de l'histogramme, telles que la définition de la largeur des barres, l'espacement entre les

barres, les marges autour de l'histogramme, les étiquettes des axes et le titre. De plus, elle est capable d'ajuster automatiquement la taille des barres en fonction de la plage des données, assurant ainsi une représentation visuelle optimale des données fournies.

Ci-dessous est présentée l'interface graphique de la classe Histogram :

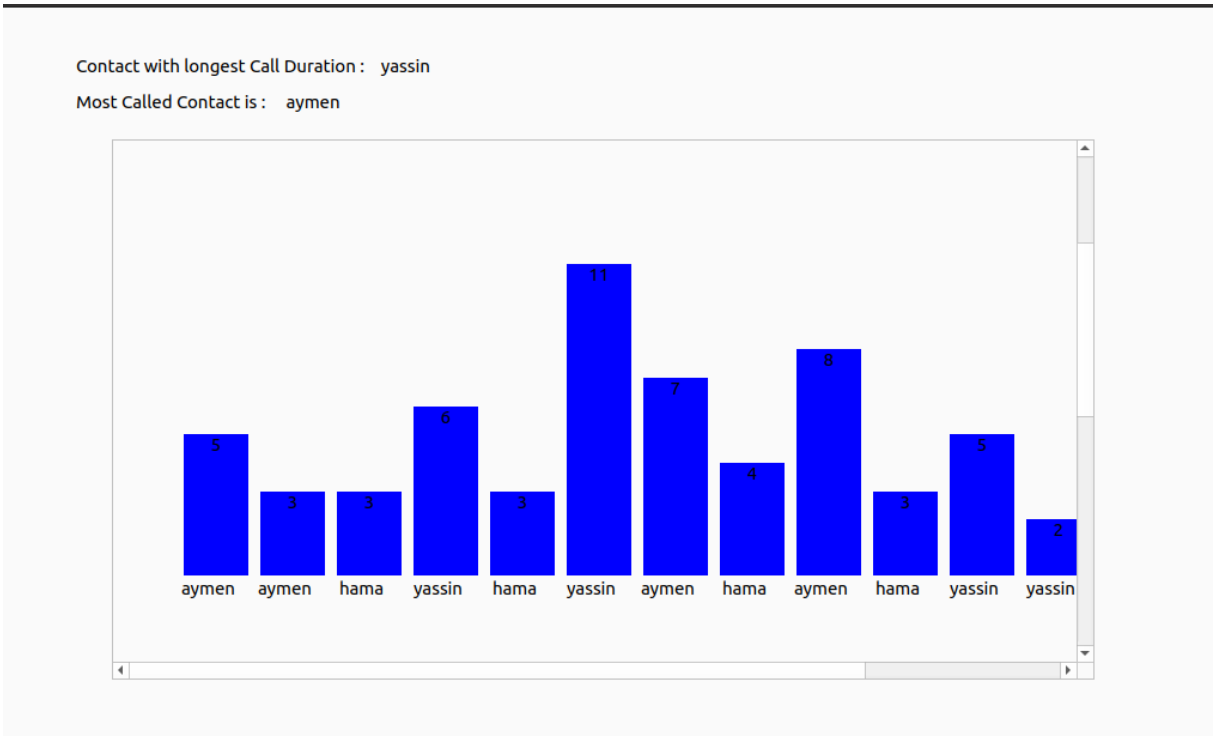

FIGURE 2.9 – Histogramme Dialog

2.3.7 Classe Stat

La classe 'Stat' est un composant crucial de l'application, permettant à l'utilisateur d'accéder aux informations statistiques sur le nombre d'appels, le nombre de contacts, ainsi que la durée moyenne des appels.

Elle offre une interface conviviale présentant ces statistiques de manière claire et compréhensible pour l'utilisateur.

Ci-dessous est présentée l'interface graphique de la classe Stat :

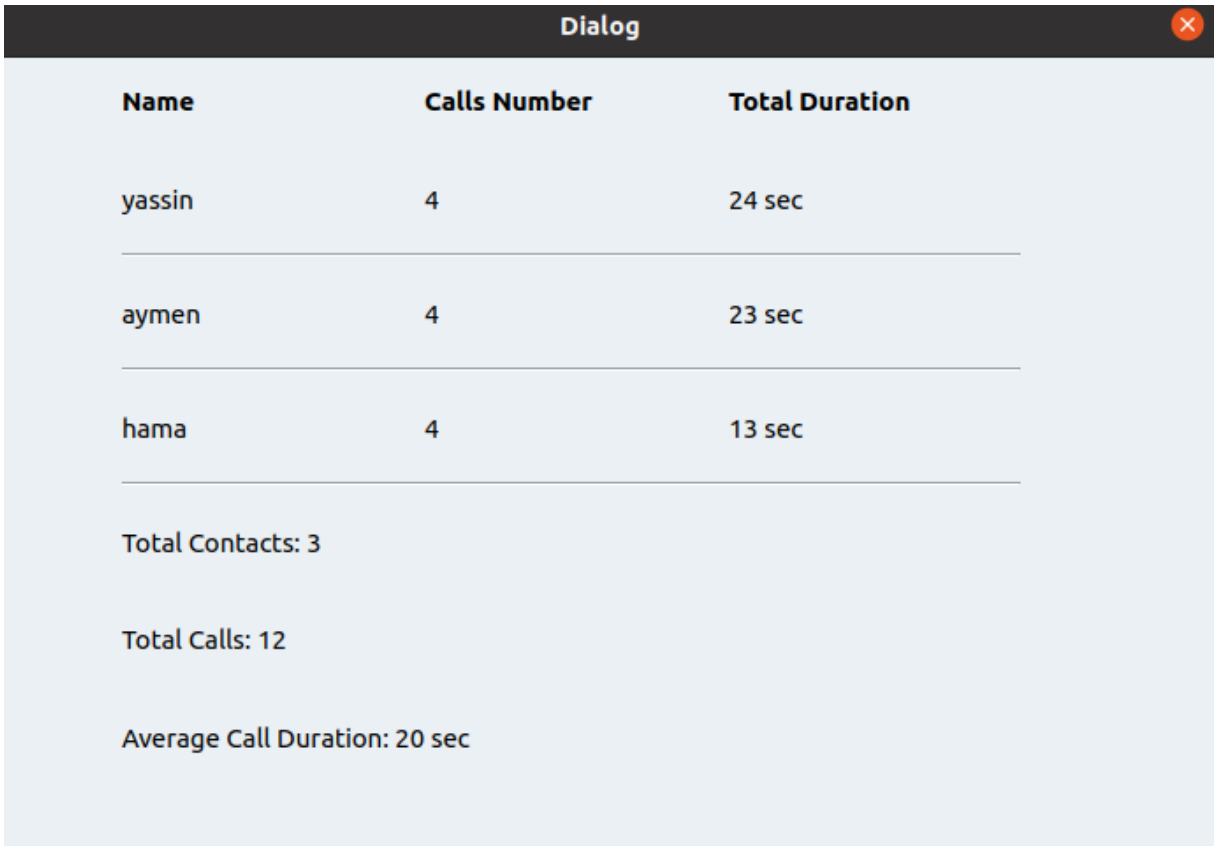

FIGURE 2.10 – Statistiques Dialog

En résumé, la classe Stat fournit un aperçu statistique complet de l'activité des appels et des contacts dans l'application, permettant aux utilisateurs d'analyser rapidement et facilement leurs habitudes de communication.

2.4 Conclusion

Ce chapitre a exposé en détail la réalisation du projet à travers l'implémentation des différentes classes et fonctionnalités nécessaires à notre application. Chaque composant a été examiné minutieusement, mettant en lumière son rôle dans le fonctionnement global de l'application.

Grâce à ces efforts de développement, une application fonctionnelle et conviviale pour la gestion des contacts et des appels téléphoniques a été créée.

Chapitre 3

Conclusion générale

Ce projet a abouti à une application de gestion de contacts et d'appels efficace et conviviale, répondant aux besoins des utilisateurs en matière de communication.

Les différentes fonctionnalités implémentées, telles que l'ajout, la modification, la suppression et la recherche de contacts selon des critères spécifiques, ainsi que la gestion des appels avec la possibilité de répondre ou de refuser, ont permis une utilisation intuitive de l'application.

De plus, la génération d'un histogramme et de statistiques basées sur les données d'appels a offert aux utilisateurs une visualisation claire de leurs habitudes de communication, facilitant ainsi leur analyse et leur interprétation.

L'utilisation de la programmation orientée objet avec C++ et l'intégration de la bibliothèque Qt ont permis la conception d'une interface graphique attrayante et fonctionnelle. La modularité du code, avec la création de classes dédiées à la gestion de la base de données et des fonctionnalités spécifiques, facilite la maintenance et l'extension de l'application.

En outre, le projet a souligné l'importance d'une gestion efficace des connexions à la base de données SQLite, ainsi que des opérations de requête et de manipulation des données. L'utilisation de structures de données appropriées et de bonnes pratiques de programmation a garanti la fiabilité et la stabilité de l'application.

En résumé, ce projet a été une opportunité d'appliquer et de renforcer les concepts de programmation orientée objet, de développement d'interfaces utilisateur et de manipulation de données dans un contexte pratique. Il a également démontré l'importance de concevoir des applications centrées sur l'utilisateur, offrant des fonctionnalités efficaces et une expérience utilisateur optimale.