Université De Fianarantsoa

Ecole Nationale d'Informatique

Mention : Informatique

Parcours : Informatique Générale

Rapport de projet fin d'année en passage en deuxième année

Intitulé

Conception et Réalisation d'une application pour la Gestion de matériels informatique

Par: Mr. RAKOTOMAMINIRINA Fanomezaniavo F Sarobidy

Mr. NOMENJANAHARY Martela Bruno

Mr. MBOLA Tsimahory Tolivoatse

Mr ANDRIANIAINA Florin Bichat

Mr. RAZAFIMANANDROVOLA Jonathan

Présenté le : 30 octobre 2024

Encadreur : Docteur RAKOTOASIMBAHOAKA Cyprien Robert

Année académique : 2023 – 2024

CURRICULUM VITAE

PROFIL

- RAKOTOMAMINIRIANA FANOMEZANIAVO FENOHERY SAROBIDY
- Age : 22 ans
- Sexe : Masculin
- Situation matrimoniale : Célibataire
- Téléphone : +261342875234
- E-mail : rfanomezaniavo@gmail.com
- Adresse : Maninday

CURSUS

- 2023-2024 : Première année de licence professionnelle en INFORMATIQUE GÉNÉRALE À L'Ecole Nationale d'Informatique
- 2022-2023 : Première année de licence Faculté science à L'Université Ankatso parcours MI
- 2019-2020 : Diplôme Baccalauréat série C

Compétences en INFORMATIQUES

- Informatique Bureautique (Word, Excel, PowerPoint)
- Programmation web (HTML, CSS, JavaScript)
- Base en programmation orienté objet
- Notions sur les systèmes d'exploitation Windows et Linux
- Notions en réseaux informatiques

Compétences linguistiques

- Malgache : langue maternelle
- Français : Intermédiaire
- Anglais : Débutant

Loisirs et préoccupations

- Films, jeux vidéo, graphisme et cuisine

Profil

Nom : NOMENJANAHARY
Prénom : Martela Bruno
Age : 25
Situation matrimoniale : Célibataire
Téléphone : 038 95 442 24
E-mail : martelabruno@7gmail.com
Adresse : Maninday Toliara

Cursus

- 2023-2024 : Première année de licence professionnelle en Informatique Générale à L'Ecole Nationale d'Informatique
- 2022-2023 : Diplôme du Baccalauréat, série Scientifique, option D

Compétences en Informatique

- Informatique bureautique (Word, Excell)
- Notions sur les systèmes d'exploitation Windows et Linux
- Notions de base en systèmes et réseaux
- Programmation web (HTML, CSS, JS)

Competences Linguistiques

- Jeux video
- Animation Chinoises

Profil

- Nom : MBOLA
- Prénom : Tsimahory Tolivoatse
- Age : 20
- Situation matrimoniale :
- Téléphone : 033 01 930 42
- E-mail : mbolatsima@gmail.com
- Adresse : Maninday Toliara

Cursus

- 2023-2024 : Première année de licence professionnelle en Informatique Générale à L'Ecole Nationale d'Informatique
- 2022-2023 : Diplôme du Baccalauréat, série Scientifique, option D

Compétences en Informatique

- Informatique bureautique (Word, Excell)
- Notions sur les systèmes d'exploitation Windows et Linux
- Notions de base en systèmes et réseaux
- Programmation web (HTML, CSS, JS)

Competences Linguistiques

- Jeux video
- Animation Japonaise

Profil

- Nom : ANDRIANAINA
- Prénom : F lorin Bichat
- Age : 21
- Situation matrimoniale : Célibataire
- Téléphone : 034 77 678 52
- E-mail : andriainiainabicha@gmail.com
- Adresse : Mahavatse Toliara

Cursus

- 2023-2024 : Première année de licence professionnelle en Informatique Générale à L'Ecole Nationale d'Informatique
- 2020-2021 : Diplôme du Baccalauréat, série Scientifique, option D

Compétences en Informatique

- Informatique bureautique (Word, Excell)
- Notions sur les systèmes d'exploitation Windows et Linux
- Notions de base en systèmes et réseaux
- Programmation web (HTML,CSS,JS)

Competences Lingistiques

- Jeux video
- Dessin

Profil

- Nom : RAZAFIMANANDROVOLA
- Prénom : Jonatan
- Age : 19
- Situation matrimoniale : Célibataire
- Téléphone : 033 60 040 08
- E-mail : fahasoavanaemm@gmail.com
- Adresse : Maninday Toliara

Cursus

- 2023-2024 : Première année de licence professionnelle en Informatique Générale à l'Ecole Nationale d'Informatique
- 2022-2023 : Diplôme du Baccalauréat, série Scientifique, option D

Compétences en Informatique

- Informatique bureautique (Word, Excell)
- Notions sur les systèmes d'exploitation Windows et Linux
- Notions de base en systèmes et réseaux
- Programmation web (HTML, CSS, JS)

Compétences Linguistiques

- Jeux vidéo
- Musicien

REMERCIEMENTS

En introduction à ce rapport, nous souhaitons exprimer notre profonde gratitude

Au Docteur HDR Thomas Mahatody ; Actuel Président de l'université de Fianarantsoa qui dirige avec bienveillance l'université de Fianarantsoa

Mr RAZAFINDRAMONJA Herinantenaina Clément Aubert Assistant d'Enseignement Supérieur à la recherche Professeur à l'ENI, pour sa générosité de nous avoir partagé son enseignement malgré ses problèmes et ses autres obligation professionnelles

A Monsieur RAKOTOASIMBAHOAKA Cyprien Robert, Maitre de conférences pour son encadrement. On le remercie pour la qualité de sa travaille ainsi que sa patiente, sa rigueur et sa disponibilité durant la préparation de ce projet

A tout le corps enseignant pour la générosité et la patiente don ils ont pu faire preuve malgré leurs charges académiques et professionnelles

A nos parents qui nous ont soutenus financièrement et moralement tout au long de l'année

Et enfin nos amis avec qui on s'est entraide et qui nous ont aussi soutenu moralement

Listes des figures :

Listes des tableaux :

Sommaire

Introduction

Dans le monde d'aujourd'hui, la technologie se propageant Rapidement par l'évolution des matériels Informatique. Face à cette situation, toutes les entreprises et les organisations sont confrontées A des défis complexes en lien avec la vérification et l'amélioration des outils et dispositifs. Le chemin positif de ces nouvelles technologies est nécessaire pour certifier le bon déroulement des opérations quotidienne, tout en même un accès mobile pour acheminer les matériaux nécessaires.

Pourtant, s'administrer à la main, de ces notaires à travers laboratoire surtout pour les grandes structures disposant d'un garage informatique inclut. La grande quantité de données aussi bien et domaines géographie pouvant constituer une solution efficace. Face à cette problématique, la question principale devient alors : quelles caractéristiques et fonctionnalités principalement à une formation pour l'organisation tout en veillant aussi dans la gestion la gestion de leurs équipements informatique ?

Pour mieux répondre à cette question, nous proposons une bonne structure au nom breveté par une génération intellectuel beaucoup organisationnel et projet ou bien élaboré. Ensuite, nous examinerons les enjeux spécifiques à la création d'une application des matériels informatique par la suite. Nous aborderons la phase de conception de l'application, en mettant en lumière les choix technologiques et les principes directeurs.

Enfin, nous explorons les fonctionnalités principales de l'application en soulignant ses avantages pour la gestion des équipements informatique au sein des organisation

Partie I .

Présentation de l’ENI

Chapitre 1 : Présentation de l'Ecole Normale d'Informatique

Bienvenue à tous pour cette présentation de l'Ecole Nationale d'Informatique mieux connu comme l'ENI Madagascar. Un lieu d'apprentissage où l'on forme les ingénieurs et experts en informatique de demain.

1.1. Information d'ordre générale

L'Ecole Nationale d'Informatique, en abrégé ENI, est un établissement d'enseignement supérieur rattaché académiquement et administrativement à l'Université de Fianarantsoa. Le siège de l'Ecole se trouve à Tanambao- Antaninarenina à Fianarantsoa.

Les coordonnées pour la prise de contact avec l'Ecole sont les suivantes :

Ecole Nationale d'Informatique (ENI)
Antaninarenina Tanambao
BP 1487 FIANARANTSOA (301)
Téléphones : 034 05 733 36 ou 033 42 302 02
Adresse mail : eni@eni-univ-fianar.mg
Site : www.eni-univ-fianar.mg

1.2. Missions et historiques :

L'ENI se positionne sur l'échiquier socio-éducatif malgache comme étant le plus puissant secteur de diffusion et de vulgarisation des connaissances et des technologies informatiques.

Cette Ecole Supérieure peut être considérée aujourd'hui comme la vitrine et la pépinière des élites informaticiennes du pays.

L'Ecole s'est constituée de façon progressive au sein du (CUR) par le décret N° 83-185 du 24 Mai 1983, comme étant le seul établissement Universitaire Professionnalisé au niveau national, destiné à former des techniciens et des Ingénieurs de haut niveau, aptes à répondre aux besoins et exigences d'Informatisation des entreprises, des sociétés et des organes implantés à Madagascar.

L'ENI a pour conséquent pour mission de former des spécialistes informaticiens compétents et opérationnels de différent niveau notamment :

En fournissant à des étudiants des connaissances de base en informatique ;

En leur transmettant le savoir – faire requis, à travers la professionnalisation des formations dispensées et en essayant une meilleure adéquation des formations par rapport aux besoins évolutifs des sociétés et des entreprises.

En initiant les étudiants aux activités de recherche dans les différents domaines des Technologies de l'information et de la communication (TIC).

L'implantation de cette Ecole Supérieur de technologie de pointe dans un pays en développement et dans une Province (ou Faritany) à tissu économique et industriel faiblement développé ne l'a pourtant pas défavorisée, ni empêchée des spécialistes informaticiens de bon niveau, qui sont recherchés par les entreprises, les sociétés et les organismes publics et privés sur le marché de l'emploi.

La filière de formation d'Analystes Programmeurs a été mise en place à l'Ecole en 1983, et a été gelée par la suite en 1996, tandis que la filière de formation d'ingénieure a été ouverte à l'Ecole en 1986.

Dans le cadre du programme de renforcement en l'Enseignement Supérieur (PRESUP) la filière de formation des Techniciens Supérieurs en Maintenance des Système des informatiques a été mise en place en 1986 grâce à l'appui matériel et financier de la Mission Française de coopération auprès de l'Ambassade de France à Madagascar.

Une formation pour l'obtention de la certification CCNA et / ou NETWORK + appelée « CISCO Networking Académie » a été créée à l'Ecole en 2002-2003 grâce au partenariat avec CISCO SYSTEM et l'Ecole Supérieure Polytechnique d'Antananarivo (ESPA). Cependant, cette formation n'avait pas duré longtemps.

L'organisation de stage en entreprise continue non seulement à renforcer la professionnalisation des formations dispensées, mais elle continue surtout à accroitre de façon exceptionnelle les opportunités d'embauche pour les diplômes de l'Ecole :

- *Génie Logiciel et base de Données.*
- *Administration des Système et réseaux.*

La mise en place à l'Ecole de ces deux options de formation devait répondre au besoin de basculement vers le système Licence – Master – Doctorat (LMD).

Mais la filière de formation des Techniques Supérieurs en Maintenance des Systèmes Informatiques a été gelée en 2009.

En vue de surmonter les difficultés de limitation de l'effectif des étudiants accueilles à l'Ecole, notamment à cause du manque d'infrastructure, un système de « Formation Hybride » a été mise en place à partir de l'année 2010. Il s'agit en effet d'un système de formation semi-présentielle et à distance avec l'utilisation de la visioconférence pour la formation à distance.

Le système de formation hybride a été ainsi créé à Fianarantsoa ainsi qu'à Université de Toliara.

1.3. Organigramme institutionnel de l'ENI :

Cet organigramme de l'Eole est inspiré des dispositions du décret N° 83-185 du 24 Mai 1983.

Sur cet organigramme, l'Ecole placée sous la tutelle académique et administrative de l'Université de Fianarantsoa, est dirigée par un Directeur élu par les Enseignants – Chercheurs permanents de l'Etablissement et nommé par un décret pris en Conseil des Ministres pour un mandat de 3 ans.

Le conseil de l'Ecole est l'organe délibérant de l'Ecole.

Le Collège des Enseignants regroupent tous les enseignants-chercheurs de l'Ecole est chargé de résoudre les problèmes liés à l'organisation pédagogique des enseignements. Il propose et coordonne les programmes d'activités pédagogiques.

Le conseil Scientifique propose les orientations pédagogiques et scientifiques de l'établissement, en tenant compte notamment de l'évolution du marché de travail et de l'adéquation des formations dispensées par rapport aux besoins des entreprises. Il coordonne les programmes de recherche à mettre en œuvre à l'Ecole.

Le Secrétariat principal coordonne les activités des services administratifs (Scolarité, Compatibilité, et Intendance).

Conformément aux textes en vigueur régissant les Etablissements malgaches d'Enseignement Supérieur, qui sont barrés sur le système LMD, les Départements de Formation pédagogique ont été ainsi remplacés par des Mentions et des parcours. Et les chefs des Départements ont été ainsi remplacés par des responsables des mentions et les responsables des parcours.

L'organigramme de l'Ecole Nationale d'Informatique est présenté par la Figure 1.

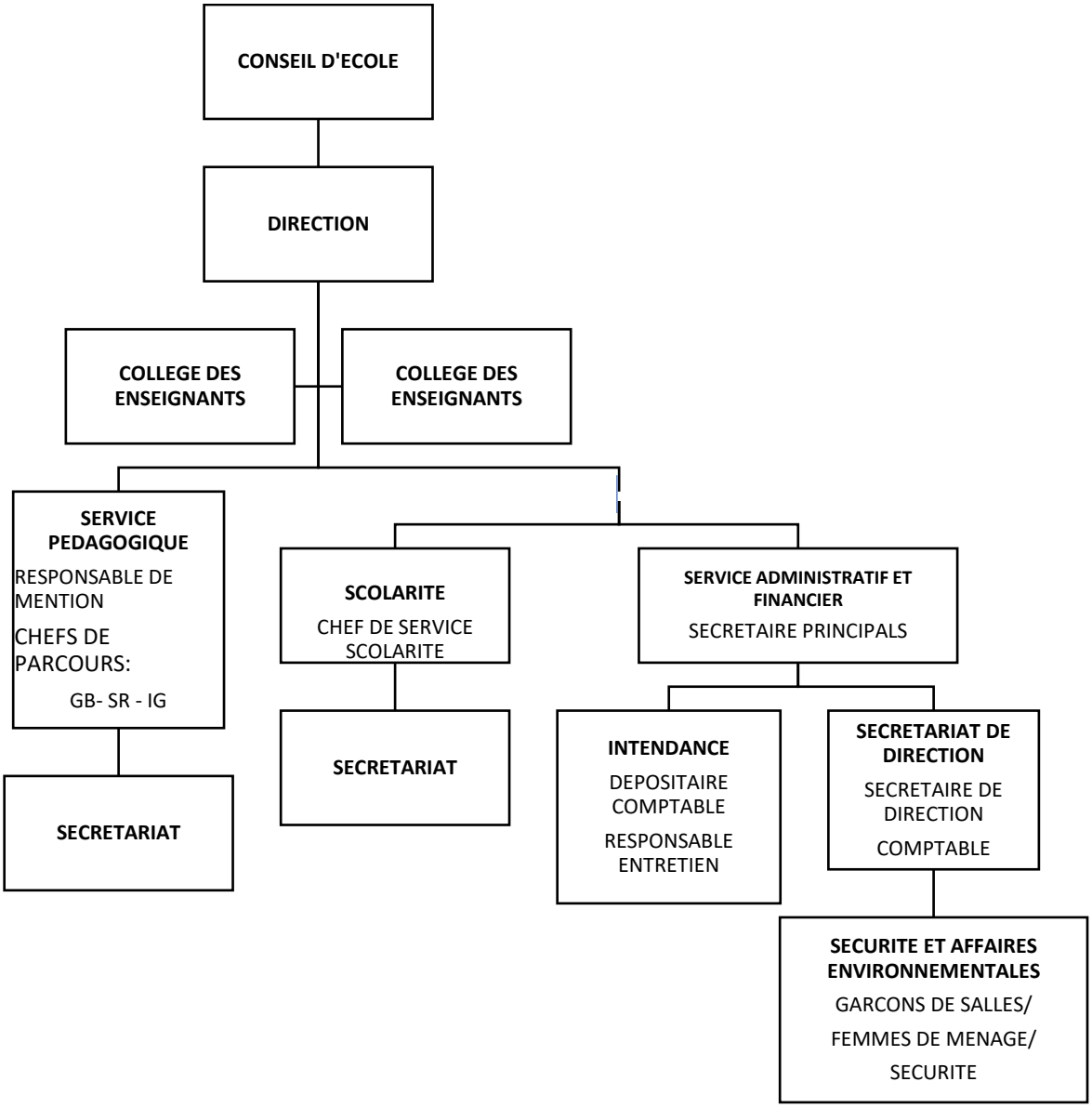
CONSEIL D'ECOLE
DIRECTION
COLLEGE DES ENSEIGNANTS
COLLEGE DES ENSEIGNANTS
SERVICE PEDAGOGIQUE
RESPONSABLE DE MENTION
CHEFS DE PARCOURS:
GB- SR - IG
SECRETARIAT
SCOLARITE
CHEF DE SERVICE SCOLARITE
SECRETARIAT
SERVICE ADMINISTRATIF ET FINANCIER
SECRETAIRE PRINCIPALS
INTENDANCE
DEPOSITAIRE COMPTABLE
RESPONSABLE ENTRETIEN
SECRETARIAT DE DIRECTION
SECRETAIRE DE DIRECTION
COMPTABLE
SECURITE ET AFFAIRES ENVIRONNEMENTALES
GARCONS DE SALLES/
FEMMES DE MENAGE/
SECURITE

Fig. 1

Chapitre 2 : **Description du projet**

2 .1 Formulation

Dans un environnement où la gestion efficace du matériel informatique est cruciale, ce projet vise à concevoir et réaliser une **application** dédiée à la **gestion** de ce matériel. L'objectif est de centraliser les informations, d'améliorer la traçabilité et de faciliter l'accès aux données.

Avec l'augmentation des équipements informatiques dans les entreprises et les établissements scolaires, la gestion efficace du matériel devient cruciale. Le suivi des inventaires, des réparations, et de l'affectation des équipements nécessite une solution numérique adaptée.

2.2 Objectifs et besoins de l'utilisateur

Objectifs Utilisateur

Suivi Efficace des Équipements : Avoir une vue d'ensemble des matériels disponibles, de leur état et de leur localisation.

Affectation et Traçabilité : Pouvoir assigner facilement des équipements à des utilisateurs spécifiques et suivre leur historique d'utilisation.

Gestion des Maintenance : Enregistrer les demandes de maintenance et suivre l'état des réparations pour garantir un matériel fonctionnel.

Analyse des Données : Accéder à des rapports et des statistiques pour optimiser l'utilisation des ressources et prévoir les besoins futurs.

Amélioration de l'Efficacité : Réduire le temps consacré à la gestion manuelle et aux recherches d'équipements.

Sécurité des Données : Protéger les informations sensibles liées aux utilisateurs et au matériel.

Besoins Utilisateur

Interface Intuitive : Une interface conviviale et facile à utiliser, permettant aux utilisateurs de naviguer sans difficulté.

Fonctionnalités de Recherche : Outils de filtrage et de recherche pour localiser rapidement un équipement spécifique.

Alertes et Notification : Recevoir des alertes pour les maintenances à venir, les équipements endommagés ou les renouvellements nécessaires.

Accès Multi-utilisateur : Permettre à différents utilisateurs de se connecter avec des niveaux d'accès adaptés (administrateurs, utilisateurs standard).

2. 3 Moyen matériel

Ces caractéristiques garantiront que l'ordinateur soit suffisamment puissant et performant pour développer et tester l'application de gestion de matériel informatique efficacement. Il est également conseillé de prévoir des sauvegardes régulières et un logiciel de sécurité pour protéger les données du projet.

Processeur

Type : Processeur multicœur (i5 ou équivalent). **Vitesse :** Minimum 2.5 GHz.

Mémoire RAM : Capacité : Au moins 8 Go (préférablement 16 Go pour un développement fluide). **:** DDR4 pour des performances optimales.

Stockage : Type : SSD pour des temps de chargement rapides. **Capacité :** Minimum 256 Go (préférablement 512 Go ou plus pour stocker des fichiers et des bases de données).

Carte Graphique : Type : Intégrée (suffisante pour le développement web) ou dédiée si des tâches graphiques lourdes sont prévues. **Mémoire :** 2 Go de VRAM minimum si dédiée.

Système d'Exploitation : Type : Windows 10/11, MacOs, ou distribution Linux (Ubuntu, Fedora, etc.) selon les préférences de développement.

Connectivité : Ports : USB 3.0/3.1, HDMI/Display Port, Ethernet, et Wi-Fi. **Bluetooth :** Pour connecter des périphériques sans fil.

Écran : Taille : Minimum 15 pouces. **Résolution :** Full HD (1920x1080) pour un confort de travail.

Clavier et Souris : Confortables et adaptés à de longues sessions de travail.

Autonomie (pour les portables) : Minimum 5-7 heures pour les ordinateurs portables, si applicable.

2.4 Résultats attendus

Pour ce qui est des résultats attendu, l'application doit avoir des trucs importants. Elle doit aider à garder un œil sur combien de matériels on a en stock, pour éviter qu'on se retrouve sans rien. Et elle doit noter chaque fois qu'on ajute ou qu'on retire des matériels, avec les dates e les quantités. Comme ça on a tout en détail dans la base de données, prêt pour des analyses futures ou si quelqu'un vient vérifie.

•*Gestion des inventaires*

L'application doit permettre un suivi facile de niveau des stocks pour éviter les ruptures des stocks et maintenir un inventaire précis et à jour.

• *Enregistrement des entrées/sorties*

L'application de gestion de stock des matériels informatiques enregistre toutes les entrées et sorties des matériels existants. Cela pour garder les traces détaillées de tous mouvements de stocks, y compris les dates et les quantités, accessible dans la base de données pour les potentielles futures analyses ou pour répondre à tout audit éventuel.

• *Recherche dans la base de données*

Cette application s'accompagne d'une fonctionnalité de recherche pour trouver rapidement les produits par toute information concernant un produit entrée dans la base de données. Cette fonctionnalité facilite la recherche d'informations spécifiques sur ces

2.5 CHRONOGRAMME

Ce diagramme permet de visualiser l'ordre des tâches, leurs durées et leurs interdépendances durant lequel on a conçu notre projet

CHRONOGRAMME DU PROJET (diagramme de Gantt)

Activités	**Dates**	**Durée**
Réception du projet	23 Septembre 2024	1 jour
Conception du MCD	24 au 1 Octobre 2024	1 semaine
Conception du MLD	2 au 6 Octobre 2024	5 jours
Conception de l'Application	7 au 30 Octobre 2024	3 semaines
Création du livre	25 au 30 Octobre 2024	1 semaine

Tab 1

PARTIE II.

Analyse et conception

Chapitre 3 : ANALYSE PREALABLE

3.1. Analyse de l'existant

Pour gérer la circulation des informations entre les acteurs, en utilisant le traitement actuel de l'application et le diagramme de flux. Il est important de mettre en place un système de gestion des flux d'information. Cela implique plusieurs étapes et éléments clés. Voici comment procéder : Identification des acteurs, Définition des processus, Création du diagramme de Flux de Données (DFD), Spécification des règles de circulation, Automatisation du traitement.

3. 2. Critique de l'existant et proposition des solutions

- Critique : Dans la zone de performances et obsolescence les composants informatiques tels que les processeurs, la RAM, et les GPU deviennent obsolètes rapidement, limitant la durée de vie des systèmes. D'autre cas, Sur la consommation d'Energie et Ecoresponsabilité les composants modernes (GPU de gaming, serveurs) est élevée, et leur production implique des ressources rares
- Solution : Opter pour des composants modulaires et facilement évolutifs
  (Comme les PC de type NUC ou les stations de travail modulaires), et
  Privilégier l'achat de matériel offrant une capacité de mise à jour importante pour prolonger la durée de vie du système. A part, choisir des composants à faible consommation d'énergie (processeurs et GPU avec des TDP plus bas). Investir dans des dispositifs écoénergétique, comme les alimentations certifiées 80 plus Gold ou Platinum.

3.3. Conception avant-projet

L'utilisation de C++ et Qt représente un choix judicieux pour construire des solutions de gestion de matériel en informatique, garantissant performance, flexibilité et satisfaction des utilisateurs.

En parlant d'outil, avant la conception de ce projet, on avait besoin de Machine (c'est logique puisque on parle d'une conception d'application), le problème c'est qu'on avait à notre disposition que 2 Machines disponibles du coup on l'a fait avec ce qu'on avait. Une application pour la Gestion des Matériels Informatiques, on nous a donner à l'avance

l'outil pour concevoir cette application qui est le Langage de programmation C++ avec le Framework Qt Creator dont on va vous parler juste après ; certes il existe d'autre Langage de programmation Orienté Objet comme Python, JavaScript, Java, Ruby, etc… , mais C++ a aussi sa propre avantage dont a va vous donner sous forme de Tableau de comparaison ci-dessous.

Langage de Programmation Orienté Objets	Caractéristiques	Avantages	Faiblesses
C++	- Orienté Objet, Procédural, Générique - Gestion mémoire flexible - Performance ***Très élevée*** - Compatibilité avec C - Populaire (Jeux, système, …).	- Large choix de bibliothèques et Framework - Très polyvalent : peut être utilisé pour des applications bas niveau et haut niveau - Performances élevées, idéal pour les applications de haut niveau.	- Complexité accrue en raison de la gestion manuelle de la mémoire - Erreurs difficiles à déboguer (fuites de mémoire) - Longtemps d'apprentissage pour bien maîtriser tous les aspects.
Java	- Orienté Objet - Gestion du mémoire Garbage Collection - Performance élevée, mais moins rapide que C++ - Populaire en Entreprise.	- Portabilité élevée grâce à la JVM (Java Virtual Machine) - Support complet pour la programmation orientée objet - Bonnes performances, même si moins rapide que C++	- Moins performant que C++ dans les applications nécessitant des ressources intensives - Syntaxe plus verbale, rendant le code parfois lourd - Pas optimal pour les scripts rapides ou les petits projets.
Python	- Multiparadigme (Objet, Fonctionnel, etc.) - Gestion du mémoire Garbage Collection - Performance Modérée - Très populaire, IA et science des données.	- Syntaxe simple et facile à apprendre - Large communauté et écosystème pour la science des données, le web et l'IA - Grand nombre de bibliothèques pour des tâches variées (web, IA, science des données).	- Moins performant, surtout pour les applications exigeant beaucoup de calcul - Gestion dynamique des types peut entraîner des erreurs au moment de l'exécution - Pas idéal pour le développement de systèmes bas niveau.

Tab 2

D’après le Tableau de comparaison ci-dessus, le choix du langage était fait puisqu’on s’est déjà familiariser avec C et on a besoin de polyvalence et de performance alors va s’y pour C++.

Parlons un peu de ce fameux Langage.

Le **C++** est un langage de programmation de bas niveau, puissant et polyvalent, créé par **Bjarne Stroustrup** dans les années 1980. Il est considéré comme une extension du C, intégrant des fonctionnalités orientées objet comme les classes et l’héritage, tout en permettant une gestion fine des ressources et de la mémoire. C++ est particulièrement apprécié pour les **applications nécessitant des performances élevées**, comme les jeux vidéo, les systèmes embarqués, et les logiciels de calcul scientifique. Sa compatibilité avec le langage C permet de réutiliser les bibliothèques C tout en exploitant les concepts modernes.

En résumé, C++ est un choix incontournable pour les projets où la **vitesse** et le **contrôle bas niveau** sont primordiaux, bien qu’il nécessite une gestion rigoureuse de la mémoire et des erreurs pour éviter les fuites et les bogues, ce que je trouve être une faiblesse pour C++.

Chapitre 4 : CONCEPTION

4.1. Description de la base de données (MCD, MLD)

Le Modèle Conceptuel de Donnée **(MLD)** et le Modèle Logique de donnée **(MLD)** pour la Gestion de Matériel Informatiques présente les données dont on va se servir durant pour que l'**application** fait ce qu'on attend de lui.

La base de données a été conçue pour répondre aux besoins suivants :

- Gestion des matériels informatiques par les Utilisateurs

- Gestion des intervenants par l'Utilisateurs

- Gestion des entretiens de matériels informatiques par l'Utilisateurs

- Visualisation de chaque intervention sur chaque matériel Informatique

- Et Édition de l'état des prestations.

Dans la première section, le **MCD** fournit une vue d'ensemble des entités et relations principales sans se préoccuper des contraintes techniques. La section suivante présente le **MLD** qui va traduire le MCD en une structure de tables et de relations, en tenant compte des spécificités techniques requises pour la base de données.

Pour bien structurer une application, que ce soit de type DESKTOP, WEB ou autres, il est primordial d'avoir une base de données bien structurer et bien définit. On a suivi ces étapes pour la création du MCD de notre application :

Premièrement, rassembler en vrac tous ce qu'on peut avoir comme donnée et ce qu'on peut utiliser ; Ensuite, rassembler celles qui peuvent se rassembler, qui ont une même **Entité** comme par exemple pour l'entité « Utilisateurs » elle peut regrouper le Nom, Matricule, et la Fonction de l'Utilisateur dans l'entreprise ; après ça, crée les tables pour chaque entité et y mettre leur **rubriques**, mettre les **cardinalités** qui varie en fonction de la relation qui existe entre les deux entités (il y a la relation père – fils qui se traduit par (1,N) – (1,1), père – père ou (1,N) – (1,N) et la relation (1,1) – (1,1)) ; Et enfin, mettre le verbe qui lie les deux ou plusieurs Entités.

L’image suivant résume tous ce qu’on vient de dire.

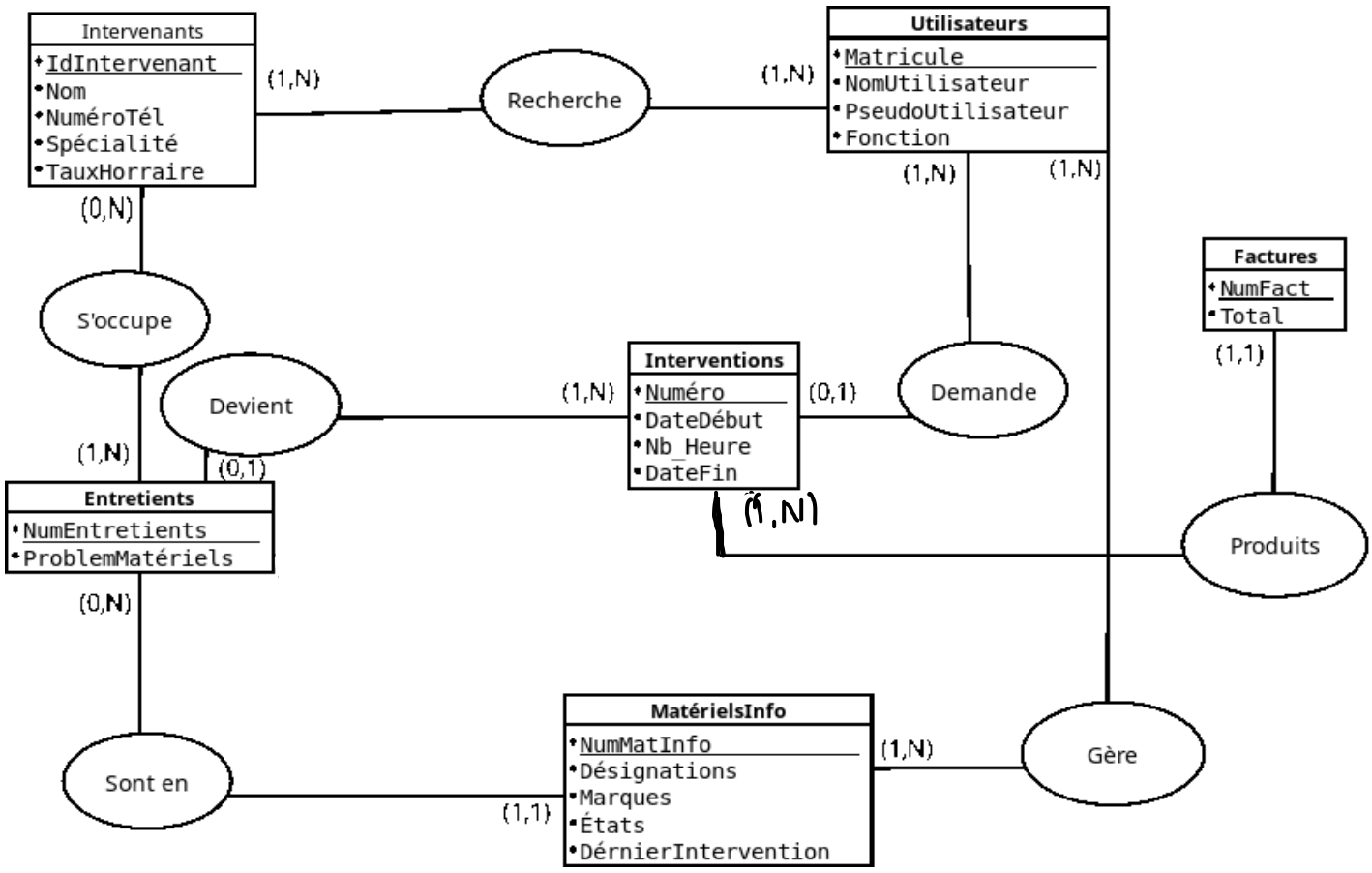

Fig. 2

Une vue plus simple de notre base de données pour pouvoir ensuite les entrées dans **SQLite** qui est ase de données qu'on va utiliser pour la conception de cette application en raison de sa simplicité.

Pour pouvoir passer du MCD au MLD il va falloir suivre certaine règle :

Premièrement, chaque Entités deviennent des **Tables**, les Rubriques deviennent des **Attributs,** mettre les **contraintes** (unicité, non-nul, etc.), chaque Tables a une **Clé primaire** qui identifie de manière unique chaque enregistrement de la table ; Après, mettre les **Clé étrangères,** c'est la relation qui lie les Tables entre elles ; A noté que chaque relation ont leurs propre caractéristique, pour la relation (1,N) – (1,N) : le verbe qui lie les deux tables deviennent une **table** qui prend les clé primaires des deux tables comme **clé étrangère,** la relation (1,N) – (1,1) : celle de gauche est le père et la clé primaire du père devient une **clé étrangère** pour le fils et la relation (1,1) – (1,1) : la clé primaire de l'une des deux tables devient une clé étrangère pour l'une des deux tables. Enfin, on peut mettre pour chaque attribut un type pour dire s'il est de type entier ou charactère, généralement c'est le passage MPD ou Modèle Physique de Donnée mais je le mets ici quand même.

Résumons tout ça avec l'image suivante

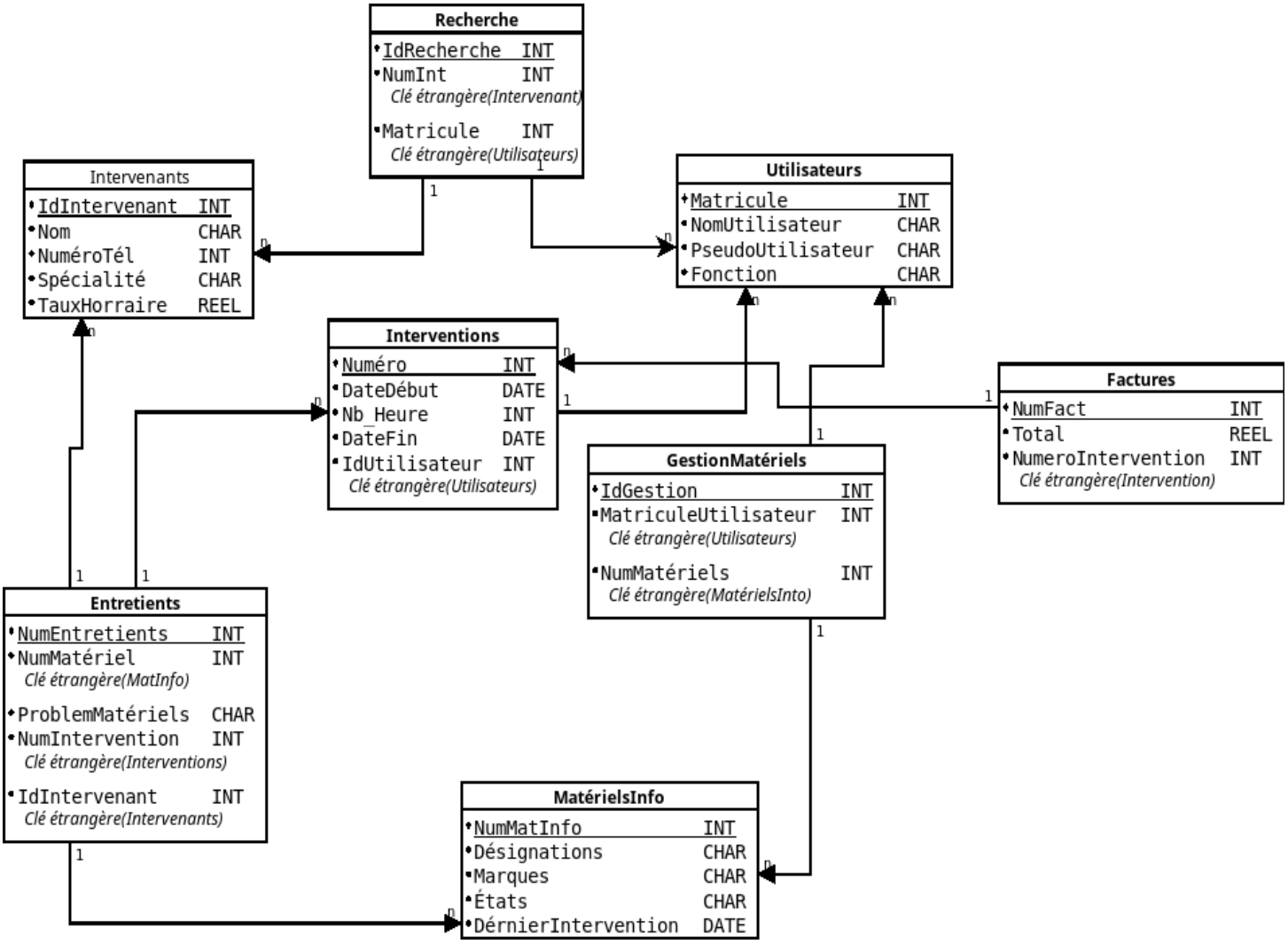

Fig. 3

4.2. Description des traitements (MCT, MOT)

Méthodes de Contrôle de Traitement (MCT)

Les MCT se concentrent sur le suivi et la régulation des traitements de matériel. Elles permettent de : **Suivre l'utilisation** des ressources matérielles (serveurs, équipements réseau, etc.) pour s'assurer qu'elles sont utilisées efficacement.

Établir des procédures de contrôle pour éviter les défaillances matérielles, en mettant en place des alertes et des diagnostics.

Optimiser les processus en identifiant les goulets d'étranglement et en ajustant les ressources en conséquence.

Méthodes d’Optimisation des Traitements (MOT)

Les MOT visent à maximiser l'efficacité et à réduire les coûts liés à la gestion du matériel. Elles incluent :

Analyse des performances : Évaluer la performance des équipements pour déterminer s'ils répondent aux besoins de l'entreprise.

Planification des capacités : Anticiper les besoins futurs en matériel en fonction de l'évolution des projets et des utilisateurs.

Gestion de la maintenance : Élaborer des plans de maintenance préventive pour prolonger la durée de vie des équipements et réduire les temps d'arrêt.

Ces traitements sont essentiels pour garantir que les infrastructures informatiques sont à la fois performantes et rentables.

PARTIE III.

REALISATION

Chapitre 5 : Mise en place de l'environnement

5.1. Installation des Outils

Comme dans tous métiers quand on va travailler il nous faut des Outils de travail.

Ici on va concevoir une Application pour la gestion de Matériels Informatiques, quand on parle de concevoir une application on parle de langage de programmation, et quand on parle de langage de programmation, on parle d'ordinateur. Tous d'abord, on a parlé du langage C++ précédemment et on a dit qu'il est très puissant et robuste, logiquement pour qu'il fonctionne normalement on aura besoin d'une machine capable de Compiler les codes qu'on va mettre dedans sans qu'il ram trop.

Pour avoir une expérience fluide, il est recommandé d'avoir au minimum une machine qui a un processeur corei5 avec 4go de ram (petit projet), une disque dur SSD serais meilleurs pour une vitesse de compilation agréable.

Après, pour la conception d'une application avec Qt Creator il faudrait avoir le logiciel, il est obligatoire d'avoir un compte si on veut installer Qt Creator en ligne (fig. 4) mais il existe aussi en version offline. On a parlé de base de données tout a l'heure alors on aura aussi besoin d'un outil pour crée notre base de données, ici j'ai DB Browser for SQLite (fig. 5) comme base de données mais vous avez le choix entre SQL ou autre ; vous pouvez aussi crée votre base de données via **des ligne de commande dans une Terminale** comme par exemple pour **SQL** :

```
CREATE DATABASE utilisateurs;

USE utilisateurs;

CREATE TABLE utilisateur ( matricule INT PRIMARY KEY, nom VARCHAR(50) NOT NULL, pseudo VARCHAR(50), fonction VARCHAR(50) );
```

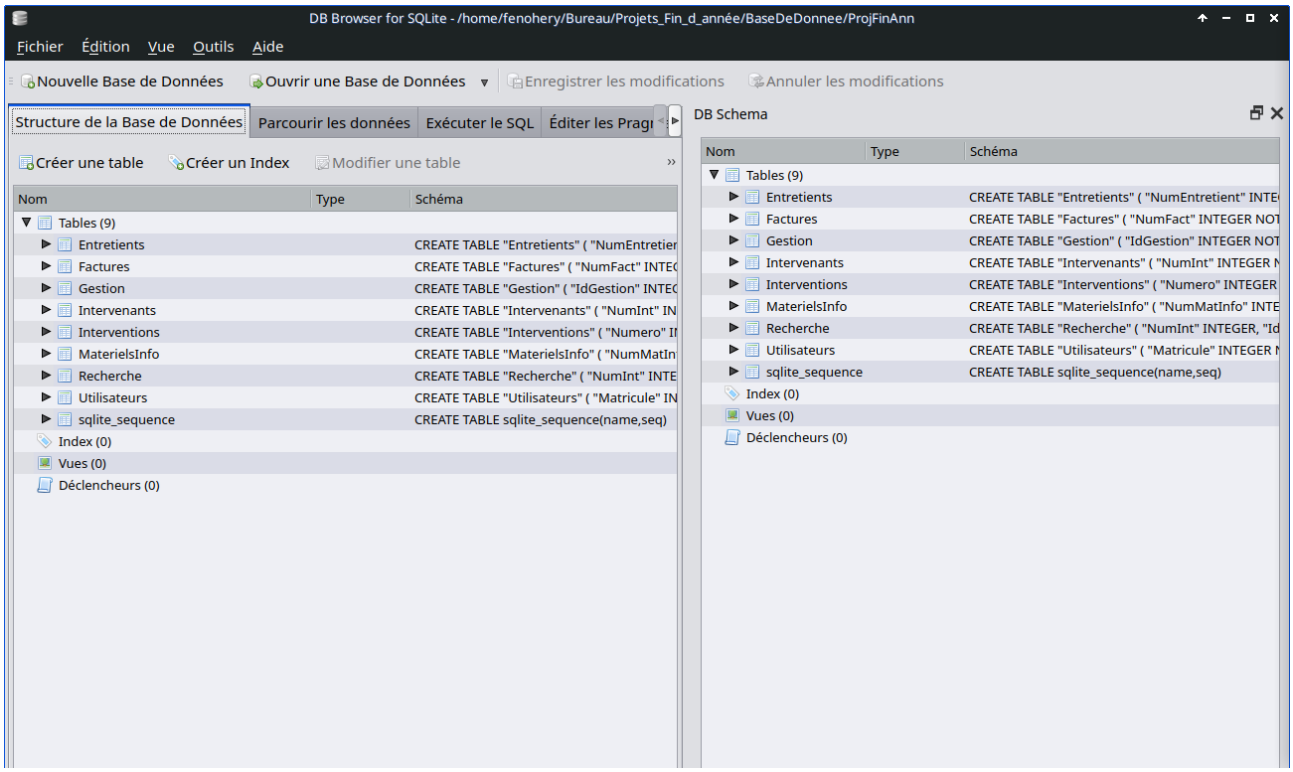
DB Browser for SQLite - /home/fenohery/Bureau/Projets_Fin_d_année/BaseDeDonnee/ProjFinAnn
Fichier Édition Vue Outils Aide
Nouvelle Base de Données
Ouvrir une Base de Données
Enregistrer les modifications
Annuler les modifications
Structure de la Base de Données
Parcourir les données
Exécuter le SQL
Éditer les Pragt
Créer une table
Créer un Index
Modifier une table
Nom
Type
Schéma
Tables (9)
Entretients
Factures
Gestion
Intervenants
Interventions
MaterielsInfo
Recherche
Utilisateurs
sqlite_sequence
Index (0)
Vues (0)
Déclencheurs (0)
DB Schema

Fig. 4

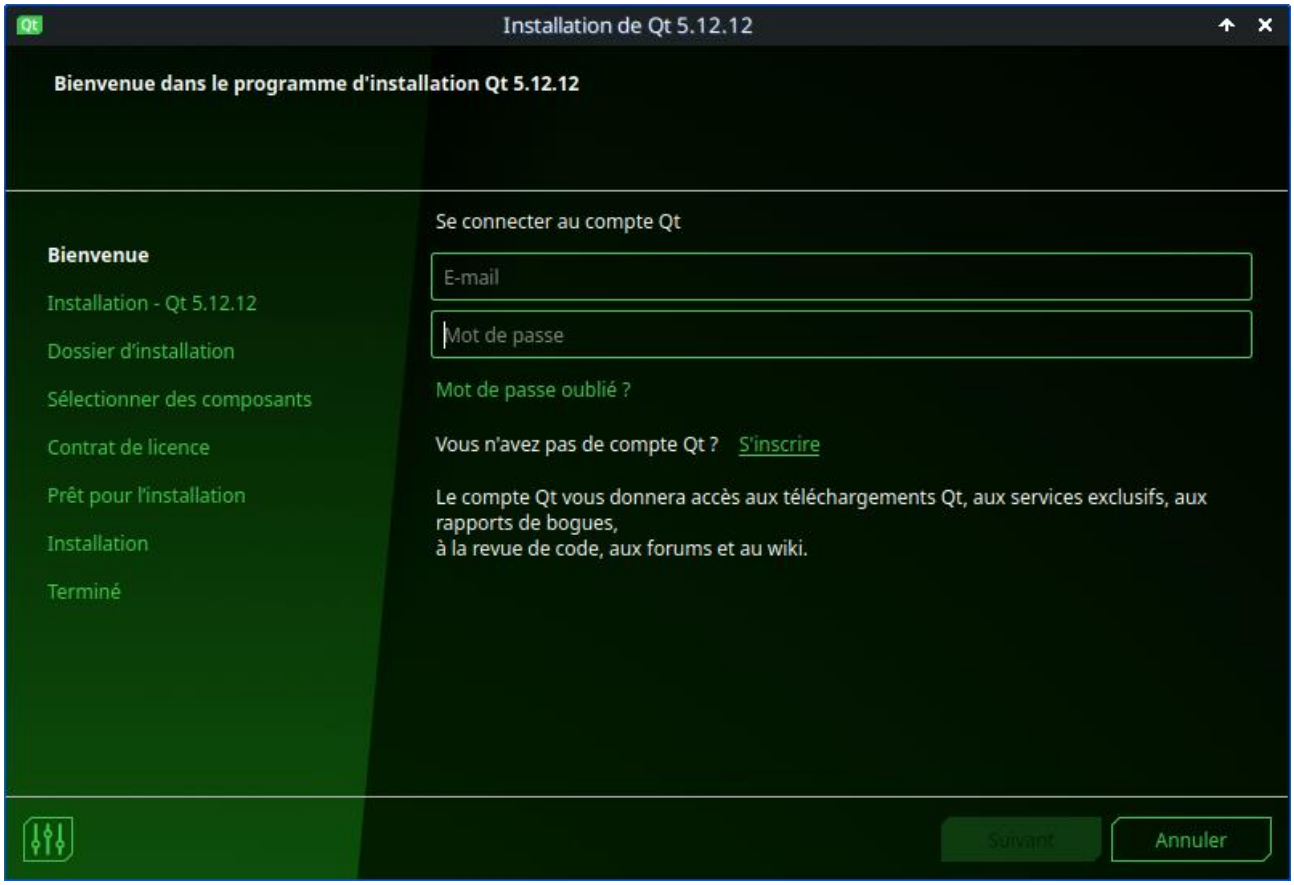
Installation de Qt 5.12.12
Bienvenue dans le programme d'installation Qt 5.12.12
Bienvenue
Installation - Qt 5.12.12
Dossier d'installation
Sélectionner des composants
Contrat de licence
Prêt pour l'installation
Installation
Terminé
Se connecter au compte Qt
E-mail
Mot de passe
Mot de passe oublié ?
Vous n'avez pas de compte Qt ? S'inscrire
Le compte Qt vous donnera accès aux téléchargements Qt, aux services exclusifs, aux rapports de bogues,
à la revue de code, aux forums et au wiki.
Suivant
Annuler

fig. 5

Une fois ces outils installer, on pourra commencer notre conception.

5.2. Configuration des Outils

Une fois les Outils installer comme il faut, configurons tout ça pour qu'il dégub notre code C++.

Une fois Qt installée en ligne ou offline, ouvrer le et nous allons créer notre Projet.

Dans l'interface de Qt il y a deux boutons nouveaux et ouvrir, on va créer notre projet alors on va dans « nouveau » une fois dedans Qt nous demande de choisir un modèle entre **« Qt widget Application »** et **« Qt Console Application »** qui est une application console, notre but et de Concevoir une application de gestion de matériel Informatique alors on va y aller dans Qt widget Application. Celle fait, il nous demande d'entrer un nom pour le Projet, pour le nôtre « projet » , et sa localisation dans notre dossier ; Ensuite dans la prochaine fenêtre choisir notre « Build system » qui sert à automatiser la compilation du code, après ça, il demande de donner l'information sur notre class, le nom, la class de base dont d'autre class pourrons hérité après ; Après, choisir si on veut traduire la langue qu'on utilise ou non, et après ça choisir le kit qu'on va utiliser qui est très important, elle varie selon la version de Qt, car c'est elle qui va déboguer notre code ; Et enfin, il nous donne une résumer de tous ce qu'on a fait et faire une retour s'il y a quelque chose à modifier.

Tous celle fait, on devrait être prêt à commencer à coder.

Notre projet est contenu dans un dossier qui est nommer avec le nom qu'on a donné au départ, il est configuré comme suit :

Il y a le fichier.pro qui va contenir tous notre projet et qui ne peut être ouvert que par Qt, pour ce projet de conception d'application on aura besoin d'une base de données, ici SQLite, alors il faut ajouter la ligne **« Qt += sql »** dans le **fichier.pro** si on veut que notre Application interagis bien avec notre base de données. Après, il les **fichiers.h** qui sont les fichiers d'en-tête où serons déclarer les variables qu'on va utiliser par la suite, et aussi c'est là-dedans qu'on va déclarer notre code pour interagir avec notre base de données :

Dans connectbdd.h :

```
#ifndef CONNECTBDD_H
#define CONNECTBDD_H

#include <QSqlDatabase>
#include <QDebug>
#include <QSqlDriver>
#include <QSqlQuery>
#include <QtSql>
#include <QSql>
#include <QSqlError>

class connectbdd
{
public:
    QSqlDatabase bdd;

    void mihidybdd() {
        bdd.close();
```

```
        qDebug() << "Mihidy donnee zao...";
        bdd.removeDatabase(QSqlDatabase::defaultConnection);
    };

    bool mivohabdd() {
//          bdd = QSqlDatabase:: addDatabase("QSQLITE", "maConnection");
        bdd = QSqlDatabase:: addDatabase("QSQLITE");

bdd.setDatabaseName("/home/fenohery/Bureau/Projets_Fin_d_année/BaseDeDonnee
/ProjFinAnn");
        QStringList drivers = QSqlDatabase::drivers();

        if (!bdd.open()) {
            qDebug() << "TSY MIVOHA BDD";
            return false;
        } else {
            qDebug() << "drivers dispo" << drivers;
            return true;
        }
    }

};

#endif // CONNECTBDD_H
```

On va juste utiliser ce fichier pour ouvrir et fermer la base de données. Après il y a les fichier.cpp où serons coder les algorithmes et fonction dont on aura besoin. Ensuite il y a le **fichier.ui** dans Forms où on pourra faire des designs comme on le veux ; et Enfin il y a les ressources où serons stoker les images, logo et tous les images dont on aura besoin.

La configuration de notre base sera faite dans DB Browser for SQLite, il n'y a pas vraiment grand-chose à configurer là-dedans, juste suivre ce qu'on a fait dans le MLD.

Chapitre 6 : DÉVELOPPEMENT DE L'APPLICATION

6.1. CRÉATION DE LA BASE DE DONNÉE

Pour bien structurer le projet on aura besoin de 6 tables qui deviendra 8 dans le MLD dis précédemment, toutes les variables de type entier serons typé par INT, les caractères TEXTE et les réels REEL.

La Table **Utilisateurs** qui va contenir le matricule, le pseudo et le Nom de l'utilisateur qui vont nous servir après dans l'application.

```
CREATE TABLE "Utilisateurs" (
        "Matricule"  INTEGER NOT NULL,
        "NomUtilisateur"    TEXT NOT NULL,
        "PseudoUtilisateur" TEXT NOT NULL,
        "Fonction"    TEXT NOT NULL,
        PRIMARY KEY("Matricule" AUTOINCREMENT)
);
```

La Tables **MatérielsInformatique** qui va contenir les matériels que l'utilisateur ajoutera.

```
CREATE TABLE "MaterielsInfo" (
        "NumMatInfo"       INTEGER NOT NULL,
        "Designations"       TEXT,
        "Marques"    TEXT,
        "Etats"        TEXT,
        "DernierIntervention"       TEXT,
        PRIMARY KEY("NumMatInfo" AUTOINCREMENT)
);
```

La Table **Intervenant** qui va contenir la liste d'intervenant dont l'utilisateur ou l'entreprise disposera.

```
CREATE TABLE "Intervenants" (
        "NumInt"     INTEGER NOT NULL,
        "Nom"        TEXT,
```

```
        "NumeroTel"TEXT UNIQUE,
        "Specialité"   TEXT,
        "TauxHorraire"      REAL,
        PRIMARY KEY("NumInt" AUTOINCREMENT)
);
```

La Table **Entretients** qui réunira la liste de matériels en entretient par un Intervenants.

```
CREATE TABLE "Entretients" (
        "NumEntretient"     INTEGER NOT NULL,
        "NumMateriel"       INTEGER,
        "ProblemMateriel"  TEXT,
        "NumeroIntervention"       INTEGER,
        "IdIntervenant"      INTEGER,
        FOREIGN KEY("NumMateriel") REFERENCES
"MaterielsInfo"("NumMatInfo"),
        FOREIGN KEY("NumeroIntervention") REFERENCES
"Interventions"("Numero"),
        FOREIGN KEY("IdIntervenant") REFERENCES "Intervenants"("NumInt"),
        PRIMARY KEY("NumEntretient" AUTOINCREMENT)
);
```

La Table **Facture** qui va contenir le **« total à payer »** en fonction de l'heure et du taux Horaires de l'Intervenants.

```
CREATE TABLE "Factures" (
        "NumFact"   INTEGER NOT NULL,
        "Total_a_payer"      REAL,
        "NumEntervention"INTEGER UNIQUE,
        FOREIGN KEY("NumEntervention") REFERENCES
"Interventions"("Numero"),
        PRIMARY KEY("NumFact" AUTOINCREMENT)
);
```

La Table **Gestion** qui va contenir la liste de matériels avec le Matricule de l'Utilisateur dont on pourra voir les détails après.

```
CREATE TABLE "Gestion" (
```

```
        "IdGestion"  INTEGER NOT NULL,
        "MatriculeUtilisateur"       INTEGER,
        "IdMatInfo" INTEGER,
        FOREIGN KEY("MatriculeUtilisateur") REFERENCES
"Utilisateurs"("Matricule"),
        FOREIGN KEY("IdMatInfo") REFERENCES
"MaterielsInfo"("NumMatInfo"),
        PRIMARY KEY("IdGestion" AUTOINCREMENT)
);
```

La Table **Interventions** qui contient des données de l'entretient de chaque Matériels.

```
CREATE TABLE "Interventions" (
        "Numero"     INTEGER NOT NULL,
        "DateDebut" INTEGER,
        "Nb_heure"  INTEGER,
        "DateFin"     INTEGER,
        "IdUtilisateur"       INTEGER,
        PRIMARY KEY("Numero" AUTOINCREMENT),
        FOREIGN KEY("IdUtilisateur") REFERENCES "Utilisateurs"("Matricule")
);
```

La Table **Recherche** qui va contenir les données que l'utilisateur tapera quand il va chercher des Intervenants.

```
CREATE TABLE "Recherche" (
        "NumInt"     INTEGER,
        "IdUtilisateur"       INTEGER,
        FOREIGN KEY("NumInt") REFERENCES "Intervenants"("NumInt"),
        FOREIGN KEY("IdUtilisateur") REFERENCES "Utilisateurs"("Matricule")
);
```

6.2. CODAGE DE L'APPLICATION

Au départ, ça nous a pris des jours pour comprendre le fonctionnement de certain code dans l'Application (ce qui a aussi retarder certain chose car on a commencé avec aucune connaissance en Qt et C++).

Tous d'abord, on a commencé par faire bosser sur l'interface Utilisateurs dont il y aura des **LineEdit** pour le Nom, Le Pseudo que l'Utilisateur choisira et le Fonction qui désignera la fonction de l'utilisateur dans l'entreprise ou il travaille.

Dans le **mainwindow.h** va y avoir les déclarations des variables, des fonctions et les différents slots que notre Application va utiliser.

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include "stylesheet.h"

#include <QMainWindow>
#include <QtSql>
#include <QSql>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
//    QSqlDatabase bdd = QSqlDatabase:: database("MaConnexion");
//    connectbdd conn;
    stylesheet styleUtil;

    QString NomUtil;
    QString PseudoUtil;
    int MatriculeUtil;
    QString FonctionUtil;

    QString nomselectionner;
//    QString pseudoselectionner;

    ~MainWindow();

private slots:
    void on_ANNULER_clicked();

    void on_VALIDER_clicked();

    void on_EditNom_textChanged(const QString &arg1);

private:

    Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
```

Ces Fonctions et slots vont être utiliser après dans **mainwindow.cpp** je vais juste donnée les codes pour les boutons VALIDER car ça prend trop de place.

```
void MainWindow:: on_VALIDER_clicked()
{
//    declaration variable champ de text
    NomUtil = ui->EditNom->text();
    PseudoUtil = ui->EditPseudo->text();
    FonctionUtil = ui->EditFonction->text();
    if (NomUtil.isEmpty()|| FonctionUtil.isEmpty() || PseudoUtil.isEmpty())
{
        QMessageBox::warning(this, "Avertissement", "VEUILLEZ REMPLIR
SVP...");
        qDebug() << "fenoy oh...";
//        connbdd.mihidybdd();
    } else {
        connectbdd connbdd;
        connbdd.mivohabdd();
        QSqlQuery query;
        query.prepare("SELECT Matricule,PseudoUtilisateur FROM Utilisateurs
WHERE PseudoUtilisateur=:pseudo");
        query.bindValue(":pseudo", PseudoUtil);
//        qDebug() << query.value(0);
//        qDebug() << query.value(1);
        if (query.exec() && query.first()) {
            int matriculeUtilisateur = query.value(0).toInt(); // matricule
SÉLÉCTIONNER

            qDebug() << matriculeUtilisateur;
            if (query.value(1).toString() > 0) {
//                QMessageBox::critical(this, "", "PSEUDO EXISTANT VEUILLEZ
SAISIR UN AUTRE NOM UTILISATEUR");

                QMessageBox mssbx;
                mssbx.setText("PSEUDO EXISTANT \n ÊTES VOUS :
"+PseudoUtil+" ?");
                QPushButton *Y = mssbx.addButton(QMessageBox::Yes);
                QPushButton *N = mssbx.addButton(QMessageBox::No);
                mssbx.exec();
                if (mssbx.clickedButton()==Y) {
                    ListMateriels *listMat = new
ListMateriels(matriculeUtilisateur);
                    this->close();
                    connbdd.mihidybdd();
                    listMat->exec();
                } else if (mssbx.clickedButton()==N) {
                    cout << "ENTRER UN NOUVEAU PSEUDO !" << endl;
                }
            } else {
                cout << "Y A RIEN ICI" << endl;
            }
        } else {
            query.prepare("INSERT INTO
Utilisateurs(NomUtilisateur,PseudoUtilisateur,Fonction) VALUES
(:Nom,:Pseudo,:Fonction);");
            query.bindValue(":Nom",NomUtil);
            query.bindValue(":Pseudo", PseudoUtil);
            query.bindValue(":Fonction",FonctionUtil);
            qDebug() << query.exec();
            MatriculeUtil = query.lastInsertId().toInt(); // définition du
matricule pour le prendre après
```

```
            if (!query.exec()) {
                        qDebug() << "Erreur lors de l'insertion :" <<
query.lastError();
                    } else {
                        qDebug() << "Utilisateurs tafiditra...";
                        connbdd.mihidybdd();
                        ListMateriels *listMat = new
ListMateriels(MatriculeUtil); // APPEL DU FENÊTRE "ListeMatériel"
PARAMÉTRER
                        this->close();
                        listMat->setModal(true);
                        listMat->exec();
                    }
        }
    }
}
```

Après ça, on a entamé la partie page d'accueil, ce qui va trouver dans la page Entretient et ce qu'on va avoir dans la page Intervenant. Dans la partie accueil, l'Utilisateur verra la liste des matériels a sa disposition, il pourra ajouter, modifier et supprimer une Matériel comme il le veut, à chaque ajout le matériel sera accompagné par le Matricule de ce dernier dont tous les Utilisateurs pourrons voir dans la partie détails.

Dans **listmateriels.cpp :**

```
void ListMateriels::on_ajoutMat_clicked() // AJOUT matériels
{
    cout << "mampidira materiel.." << endl;
    mat = new materiel(matriculeUtilisateurGlobal); // paramétrer
"matriculeGlobal dans la fenêtre materiel
    mat->setModal(true);
    mat->exec();
    afficheMateriels();
}
```

Dans **detailsmateriels.cpp** :

```
DetailsMateriels::DetailsMateriels(int matriculeU, QWidget *parent) :
    QDialog(parent),
    ui(new Ui::DetailsMateriels)
{
    ui->setupUi(this);
    this->setWindowTitle("DETAILS DES MATÉRIELS");
    resize(700,400);
    this->matriculeUt = matriculeU; // récuperation du matricule de
l'utilisateurs

    affichedetailsMat();
}
```

On a parlé d'ajout de matériel tout à l'heure ; A chaque Matériels ajouter, le Matricule de l'Utilisateur sera envoyé automatiquement dans la base de données, spécifiquement dans Gestion.

Dans **materiel.cpp :**

```
materiel::materiel(int matricule, QWidget *parent) :
    QDialog(parent),
    ui(new Ui::materiel)
{
    ui->setupUi(this);
    this->setWindowTitle("MATERIEL");
    this->matriculeMatGlobal = matricule; // Prise du valeur de "matricule"
dans la fenêtre de début
}

void materiel::on_VALIDER_clicked()
{
    connectbdd conxbdd;
    conxbdd.mivohabdd();
    QString marque = ui->marqueEdit->text();
    QString designation = ui->designationEdit->currentText();
    QString etat = ui->etatEdit->currentText();
    QString dernierInt = ui->dernierEdit->text();

    if (marque.isEmpty() || designation.isEmpty() || etat.isEmpty()) {
        QMessageBox::warning(this, "attention", "Il semble qu'il y a des
champ vide");
    } else {
        QSqlQuery matquery;
        lastIdU = matquery.lastInsertId();
        matquery.prepare("INSERT INTO MaterielsInfo(Designations, Marques,
Etats, DernierIntervention) VALUES(:type, :marque, :etat, :dernierint);");
        matquery.bindValue(":type", designation);
        matquery.bindValue(":marque", marque);
        matquery.bindValue(":etat", etat);
        matquery.bindValue(":dernierint", dernierInt);

        if (!matquery.exec()) {
                    qDebug() << "Erreur lors de l'insertion :" <<
matquery.lastError();
                } else {
                    lastIdMateriel = matquery.lastInsertId(); // prend le
dernier IdMatériel
                    qDebug() << lastIdMateriel;
                    qDebug() << "Insertion réussie dans MATÉRIELINFO OK";
                    QSqlQuery query;
                    query.prepare("INSERT INTO
Gestion(MatriculeUtilisateur,IdMatInfo) VALUES(:IdU, :IdMat);");
                    qDebug() << matriculeMatGlobal;
                    query.bindValue(":IdU", matriculeMatGlobal);
                    query.bindValue(":IdMat", lastIdMateriel);
                    if (!query.exec()) {
                        qDebug() << "Erreur :" << "tsy mandeh le insert
materiel";
                        qDebug() << "Erreur :" << query.lastError();

                    } else {
                        qDebug() << "Insertion dans GESTION parfait ";
                        cout << "Insertion dans Gestion parfait";
                        conxbdd.mihidybdd();
                        this->close();
                    }
                }
    }
}
```

L'Utilisateur aura le pouvoir de géré chaque matériel.
Dans **listmateriels.cpp :**

```
void ListMateriels::on_modifMat_clicked() // Modification Matériels
{
    modifMat modifMat;
    modifMat.getData(marque, type, etat, dernierint);
    modifMat.Nbr = modifMat.getNb(Nb);
    modifMat.exec();
    afficheMateriels();
}

void ListMateriels::on_supprimMat_clicked() // SUPPRIMER UNE MATÉRIELE INFO
{
    qDebug() << Nb;
    QMessageBox mess;
    mess.setText("SUPPRIMER !?");
    QPushButton *o = mess.addButton(QMessageBox::Yes);
    QPushButton *n = mess.addButton(QMessageBox::Cancel);
    mess.exec();
    if (mess.clickedButton()==o) {
        connectbdd connbdd;
        connbdd.mivohabdd();
        QSqlQuery suppr;
        suppr.prepare("DELETE FROM MaterielsInfo WHERE NumMatInfo=:nb;");
        suppr.bindValue(":nb", Nb);
        if (suppr.exec()) {
            qDebug() << "ligne supprimer: " << Nb;
            connbdd.mihidybdd();
            afficheMateriels();
        } else {
            qDebug() << "Erreur de suppression.";
            qDebug() << suppr.lastError();
        }

    } else if (mess.clickedButton()==n) {
        cout << "Donnée non supprimer.." << endl;
    }
}
```

Chaque modification sera envoyée et afficher dans la liste des Matériels.
Dans **modifmat.cpp :**

```
void modifMat::on_modif_clicked()
{
    connectbdd bdd;
    bdd.mivohabdd();
    Marque = ui->marqueModif->text();
    Type = ui->typeModif->currentText();
    Etat = ui->etatModif->currentText();
    DernInt = ui->intModif->text();

    if (Marque.isEmpty() || Type.isEmpty() || Etat.isEmpty() ||
DernInt.isEmpty()) {
        QMessageBox::warning(this, "attention", "Il semble qu'il y a des
champ vide");
    } else {
        QSqlQuery matquery;
```

```
        matquery.prepare("UPDATE MaterielsInfo SET
Designations=:type,Marques=:marque,Etats=:etat,DernierIntervention=:dernier
int WHERE NumMatInfo=:NumMat;");
        matquery.bindValue(":marque", Marque);
        matquery.bindValue(":type", Type);
        matquery.bindValue(":etat", Etat);
        matquery.bindValue(":dernierint", DernInt);
        matquery.bindValue(":NumMat", Nbr); // "Nbr" est une variable prise
dans listmateriel

        if (!matquery.exec()) {
                    qDebug() << "Erreur lors du MAJ :" <<
matquery.lastError();
                } else {
                    qDebug() << "Mise à jour parfait sur la ligne: " <<
Nbr;
                }

        bdd.mihidybdd();
        this->close();
        QMessageBox::warning(this, "Info", "Modification fait sur le
materiel Numéro: "+Nbr);
    }
}
```

De même dans la liste d'Intervention, l'Utilisateur aura le pouvoir de gérer la liste d'Intervention en cours **en fonction** de l'états du matériel.

Dans **listmateriels.cpp :**

```
void ListMateriels::afficheListeIntervention() // AFFICHE LA LISTE DE
MATÉRIELS EN ENTRETIENTS
{
    connectbdd connbdd;
    connbdd.mivohabdd();
    model = new QSqlQueryModel(ui->page_3);
    model->setQuery("SELECT
Entretients.NumMateriel,ProblemMateriel,MaterielsInfo.Designations,Marques,
Etats,Interventions.DateDebut,Nb_heure,DateFin,Intervenants.Nom,NumeroTel,E
ntretients.NumEntretient "
                    "FROM Entretients INNER JOIN MaterielsInfo ON
MaterielsInfo.NumMatInfo=Entretients.NumMateriel "
                    "INNER JOIN Interventions ON
Interventions.Numero=Entretients.NumeroIntervention "
                    "INNER JOIN Intervenants ON
Intervenants.NumInt=Entretients.IdIntervenant;"
                    );
    model->setHeaderData(0,Qt::Horizontal,tr("N° Matériels"));
    model->setHeaderData(1,Qt::Horizontal,tr("Problem"));
    model->setHeaderData(2,Qt::Horizontal,tr("Désignation"));
    model->setHeaderData(3,Qt::Horizontal,tr("Marques"));
    model->setHeaderData(4,Qt::Horizontal,tr("États"));
    model->setHeaderData(5,Qt::Horizontal,tr("Debut"));
    model->setHeaderData(6,Qt::Horizontal,tr("Heure"));
    model->setHeaderData(7,Qt::Horizontal,tr("Fin"));
    model->setHeaderData(8,Qt::Horizontal,tr("NomRéparateurs"));
    model->setHeaderData(9,Qt::Horizontal,tr("Téléphone"));
    model->setHeaderData(10,Qt::Horizontal,tr("Entretient N°"));
    ui->viewEntretient->setModel(model);
```

```
    connbdd.mihidybdd();
}

void ListMateriels::on_IntervenirMat_clicked() // CLICK SUR le Boutton
INTERVENIR
{
    EntretientMateriels *entretientMat = new
EntretientMateriels(matriculeUtilisateurGlobal);
    // Récuperation des données pour les mettres dans la fenêtre
    entretientMat->getdonner(type,marque,etat,Nb);

    connectbdd conbdd;
    conbdd.mivohabdd();
    QSqlQuery queryajout;
    queryajout.prepare("SELECT Nom,NumeroTel,TauxHorraire,NumInt FROM
Intervenants WHERE NumInt=?");

    if (type=="CLAVIER" || type=="SOURIS") {
        queryajout.bindValue(0,0);
        if (queryajout.exec()) {
            while (queryajout.next()) {
                numIntervenant = queryajout.value(3).toInt();
                QString Nom = queryajout.value(0).toString();
                QString Num = queryajout.value(1).toString();
                QString Taux = queryajout.value(2).toString();
                entretientMat->getinfoEntretient(Nom, Num, Taux);
                entretientMat->getIdIntervenant(numIntervenant);
            }
        } else { qDebug() << "Erreur execution requête:
"<<queryajout.lastError().text(); }
    } else if (type=="RESEAUX" || type=="PORTABLE") {
        queryajout.bindValue(0,7);
        if (queryajout.exec()) {
            while (queryajout.next()) {
                numIntervenant = queryajout.value(3).toInt();
                QString Nom = queryajout.value(0).toString();
                QString Num = queryajout.value(1).toString();
                QString Taux = queryajout.value(2).toString();
                entretientMat->getinfoEntretient(Nom, Num, Taux);
                entretientMat->getIdIntervenant(numIntervenant);
            }
        } else { qDebug() << "Erreur execution requête:
"<<queryajout.lastError().text(); }
    } else if (type=="IMPRIMANTE" || type=="ALIMENTATION") {
        queryajout.bindValue(0,8);
        if (queryajout.exec()) {
            while (queryajout.next()) {
                numIntervenant = queryajout.value(3).toInt();
                QString Nom = queryajout.value(0).toString();
                QString Num = queryajout.value(1).toString();
                QString Taux = queryajout.value(2).toString();
                entretientMat->getinfoEntretient(Nom, Num, Taux);
                entretientMat->getIdIntervenant(numIntervenant);
            }
        } else { qDebug() << "Erreur execution requête:
"<<queryajout.lastError().text(); }
    } else if (type=="ECRAN" || type=="UNITE CENTRALE") {
        queryajout.bindValue(0,9);
        if (queryajout.exec()) {
            while (queryajout.next()) {
                numIntervenant = queryajout.value(3).toInt();
```

```
                QString Nom = queryajout.value(0).toString();
                QString Num = queryajout.value(1).toString();
                QString Taux = queryajout.value(2).toString();
                entretientMat->getinfoEntretient(Nom, Num, Taux);
                entretientMat->getIdIntervenant(numIntervenant);
            }
        } else { qDebug() << "Erreur execution requête:
"<<queryajout.lastError().text(); }
    } else {
        queryajout.bindValue(0,1);
        if (queryajout.exec()) {
            while (queryajout.next()) {
                numIntervenant = queryajout.value(3).toInt();
                QString Nom = queryajout.value(0).toString();
                QString Num = queryajout.value(1).toString();
                QString Taux = queryajout.value(2).toString();
                entretientMat->getinfoEntretient(Nom, Num, Taux);
                entretientMat->getIdIntervenant(numIntervenant);
            }
        } else { qDebug() << "Erreur execution requête:
"<<queryajout.lastError().text(); }
    }

    conbdd.mihidybdd();
    entretientMat->exec();
    afficheListeIntervention();
}

void ListMateriels::on_supprimerMatEntretenu_clicked() // SUPPRIMER
MATÉRIELS ENTRETENUE
{
    qDebug() << numEntretient;
    QMessageBox mess;
    mess.setText("SUPPRIMER !?");
    QPushButton *o = mess.addButton(QMessageBox::Yes);
    QPushButton *n = mess.addButton(QMessageBox::Cancel);
    mess.exec();
    if (mess.clickedButton()==o) {
        connectbdd connbdd;
        connbdd.mivohabdd();
        QSqlQuery suppr;
        suppr.prepare("DELETE FROM Entretients WHERE NumEntretient=:num");
        suppr.bindValue(":num", numEntretient);
        if (suppr.exec()) {
            qDebug() << "ligne supprimer: " << numEntretient;
            connbdd.mihidybdd();
            afficheListeIntervention();
            QMessageBox::information(this, "Information", "ligne
"+numEntretient+" supprimer");
        } else {
            qDebug() << "Erreur de suppression.";
            qDebug() << suppr.lastError();
        }

    } else if (mess.clickedButton()==n) {
        cout << "Donnée non supprimer.." << endl;
    }
}
```

Dans la page intervenant, l'Utilisateur pourra chercher un Intervenants à partir de son **nom** ou de sa **numéro** et affiche VIDE s'il n'y a pas d'occurrence.

Dans **listmateriels.cpp :**

```
void ListMateriels::on_zonederecherche_textChanged(const QString &text)
{
    connectbdd bdd;
    bdd.mivohabdd();
    QString queryString = QString(
            "SELECT Nom, NumeroTel,Specialité,TauxHorraire "
            "FROM Intervenants "
            "WHERE Nom LIKE '%%1%' OR NumeroTel LIKE '%%1%'"
        ).arg(text);
    model->setQuery(queryString);

    if (model->rowCount() == 0) {
        ui->viewIntervenantVide->setHidden(false);
        ui->viewIntervenantVide->setText("VIDE");
        ui->viewIntervenant->setHidden(true);
    } else {
        ui->viewIntervenantVide->setHidden(true);
        ui->viewIntervenant->setHidden(false);
    }
    ui->viewIntervenant->setModel(model);
    bdd.mihidybdd();
}
```

Dans la partie liste matériel Entretenu, on pourra voir la facture et éditer l'état des prestations en PDF.

Dans **facture.cpp :**

```
void facture::on_editionPDF_clicked()
{
    connectbdd bddfact;
    bddfact.mivohabdd();
    generationPDF();
}

void facture::generationPDF()
{
    QPdfWriter writer("facture_2.pdf");
    writer.setPageSize(QPageSize(QPageSize::A4));
    writer.setResolution(300);

    QPainter painter(&writer);
    painter.setFont(QFont("Arial", 20)); // Définition du police

    // Écrire le titre
    painter.drawText(100, 100, "Facture matériels");
    painter.drawText(100, 130, "Détails des interventions :");

    QSqlQuery query;
    query.prepare("SELECT
Entretients.NumeroIntervention,IdIntervenant,Interventions.Nb_heure,Interve
nants.TauxHorraire "
```

```
                  "FROM Entretients INNER JOIN Interventions ON
Entretients.NumeroIntervention=Interventions.Numero "
                  "INNER JOIN Intervenants ON
Entretients.IdIntervenant=Intervenants.NumInt;");

    if (query.exec()) {
        int yPosition = 200; // Position verticale de départ

        // En-tête de tableau
        painter.drawText(100, yPosition, "Numéro d'Intervention");
        painter.drawText(500, yPosition, "ID Intervenant");
        painter.drawText(900, yPosition, "Heures");
        painter.drawText(1300, yPosition, "Taux");

        yPosition += 30; // Espacement après l'en-tête

        // Récupérer et afficher les données de chaque ligne
        while (query.next()) {
            int numIntervention = query.value(0).toInt();
            int idIntervenant = query.value(1).toInt();
            int nbHeure = query.value(2).toInt();
            double taux = query.value(3).toDouble();

            painter.drawText(100, yPosition,
QString::number(numIntervention));
            painter.drawText(250, yPosition,
QString::number(idIntervenant));
            painter.drawText(400, yPosition, QString::number(nbHeure));
            painter.drawText(500, yPosition, QString::number(taux));

            yPosition += 20; // Avancer vers le bas pour la prochaine ligne
        }
        painter.end();
        qDebug() << "PDF généré avec succès.";
        this->close();
    } else {
        qDebug() << "Erreur lors de la récupération des données :" <<
query.lastError();
    }
}
```

6.3. PRÉSENTATION DE L'APPLICATION

Avant de commencer, je tiens à préciser qu'en terme d'interface, on n'a pas vraiment eu ce qu'on voulait avoir à cause du manque d'expérience en terme d'interface.

- *Fenêtre Authentification*

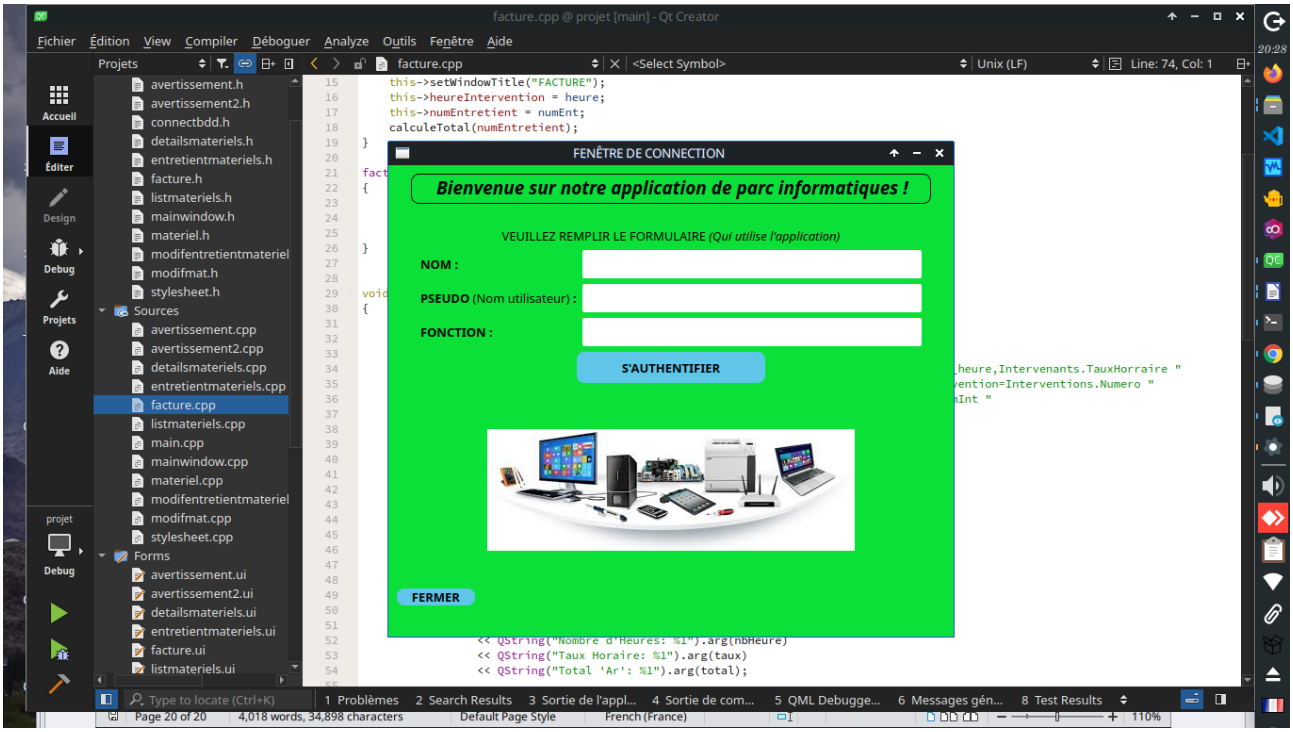

fig. 6

En tapant sur Nom, si l'Utilisateur est déjà dans la base de données, il donnera automatiquement les suggestions et si dans **pseudo** ils ont le même, l'Application renverra un message disant si c'est la même personne et utilisera le pseudo de ce dernier dans la suite.

- *Fenêtre accueil et Liste Matériels*

En entrant dans l'application, l'Utilisateurs sera accueillie par un message qui dit en haut a gauche « Bienvenue » + pseudo de l'Utilisateurs, la liste des matériels disponible avec l'Etats, la Marque et la date de son dernier entretient

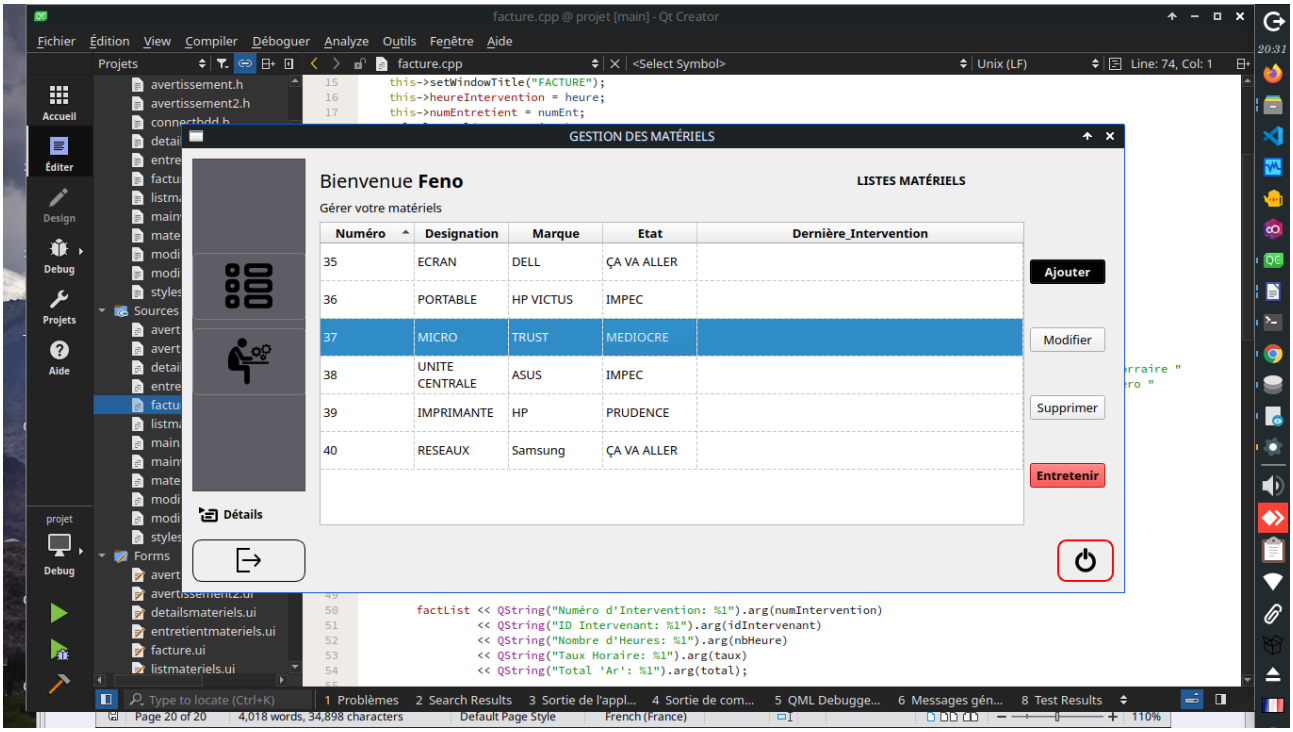

Fig. 7

- *Fenêtre Ajout Matériels*

On peut ajouter une Matériels avec la marque qu'on veut et entrer la date de la dernière entretient qui est facultatif, le matériel sera affiché immédiatement dans la liste

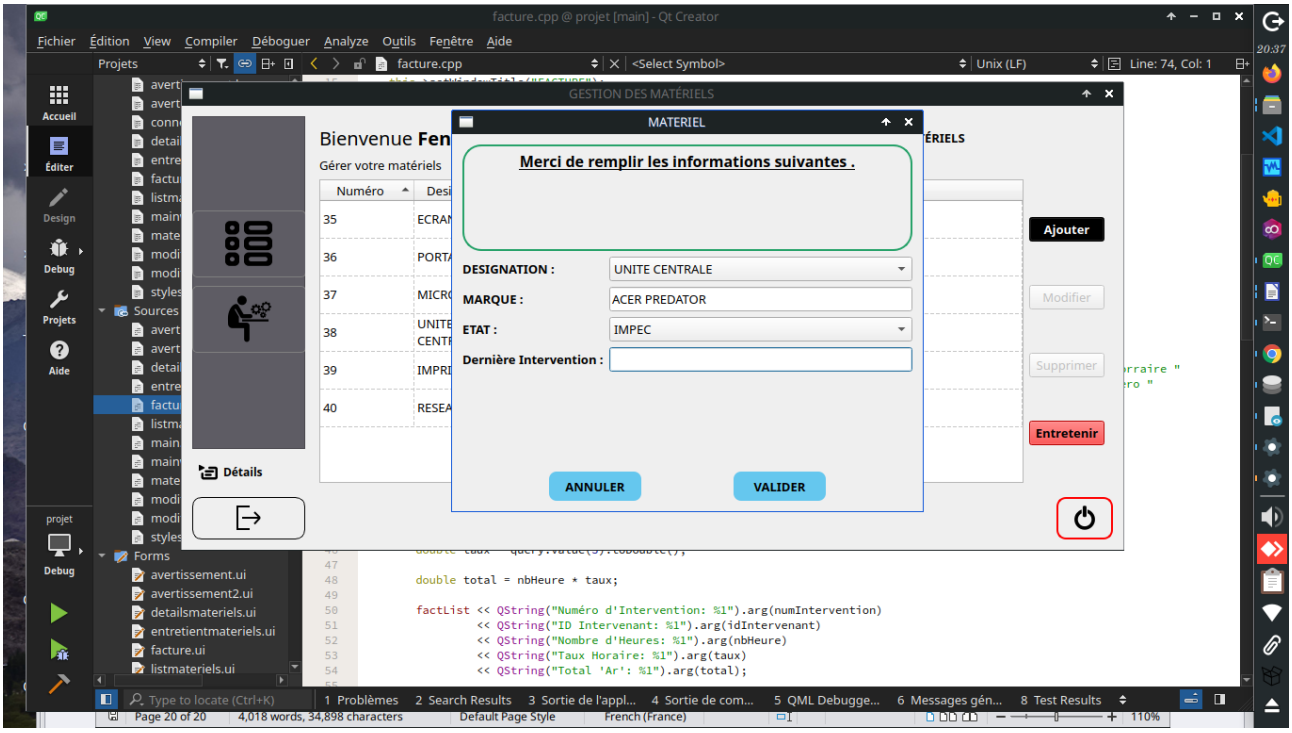

Fig. 8

- *Fenêtre Intervenants*

En effet, on peut voir la liste d'intervenant disponible dans une fenêtre et aussi faire une recherche en fonction de son Nom ou son Numéro de Téléphone

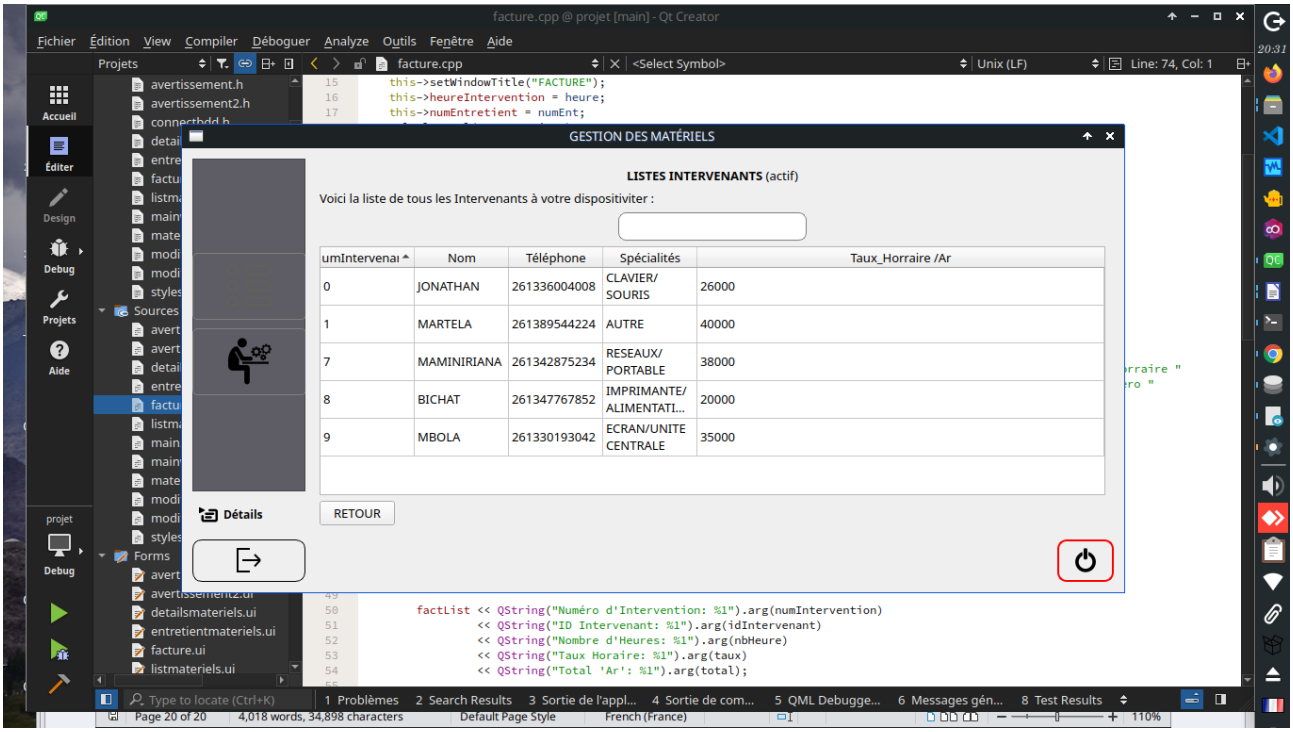

Fig. 9

- *Fenêtre Entretients et Intervention*

Dans cette fenêtre sont lister les interventions en cours, c'est a dire que l'Utilisateur click sur une Matériels et peut faire ou demander une intervention en fonction de l'Etats du Matériel. Une fois les Informations entrer il peut envoyer cette demande et il va s'afficher dans la liste de Matériels Entretenue

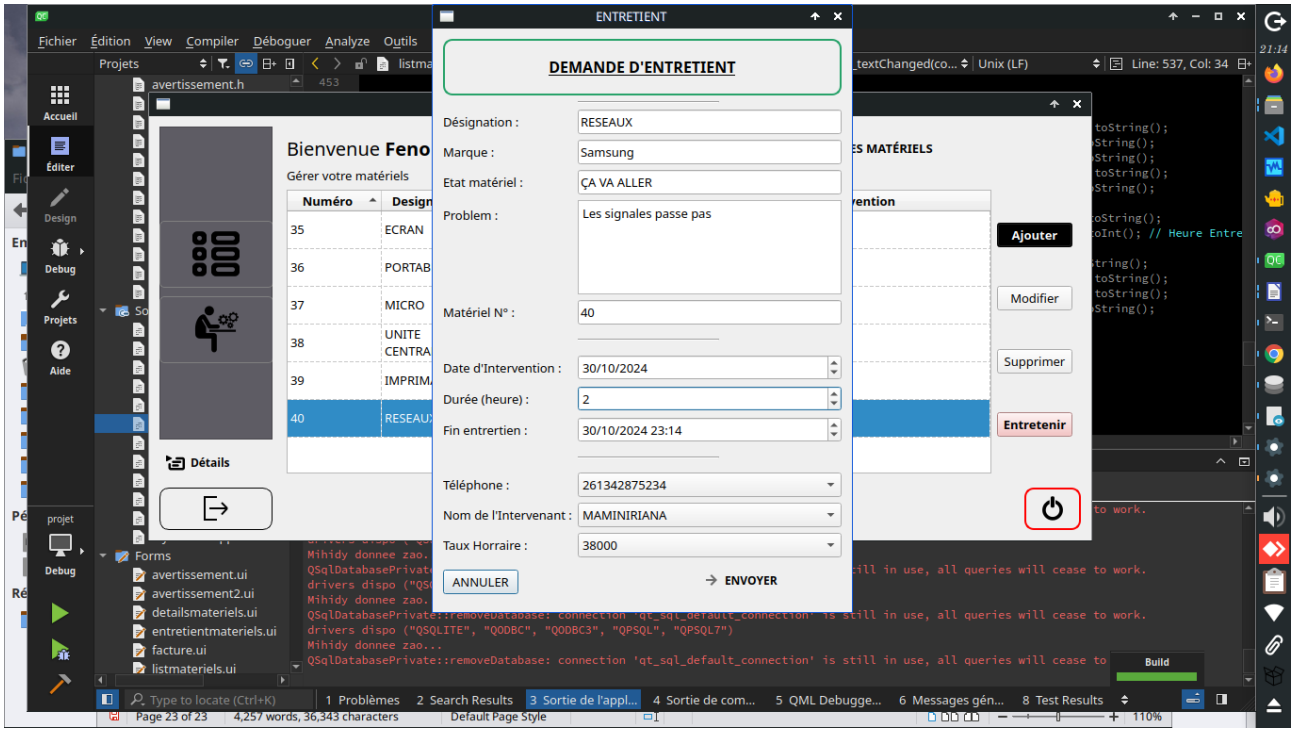

Fig.10

Après

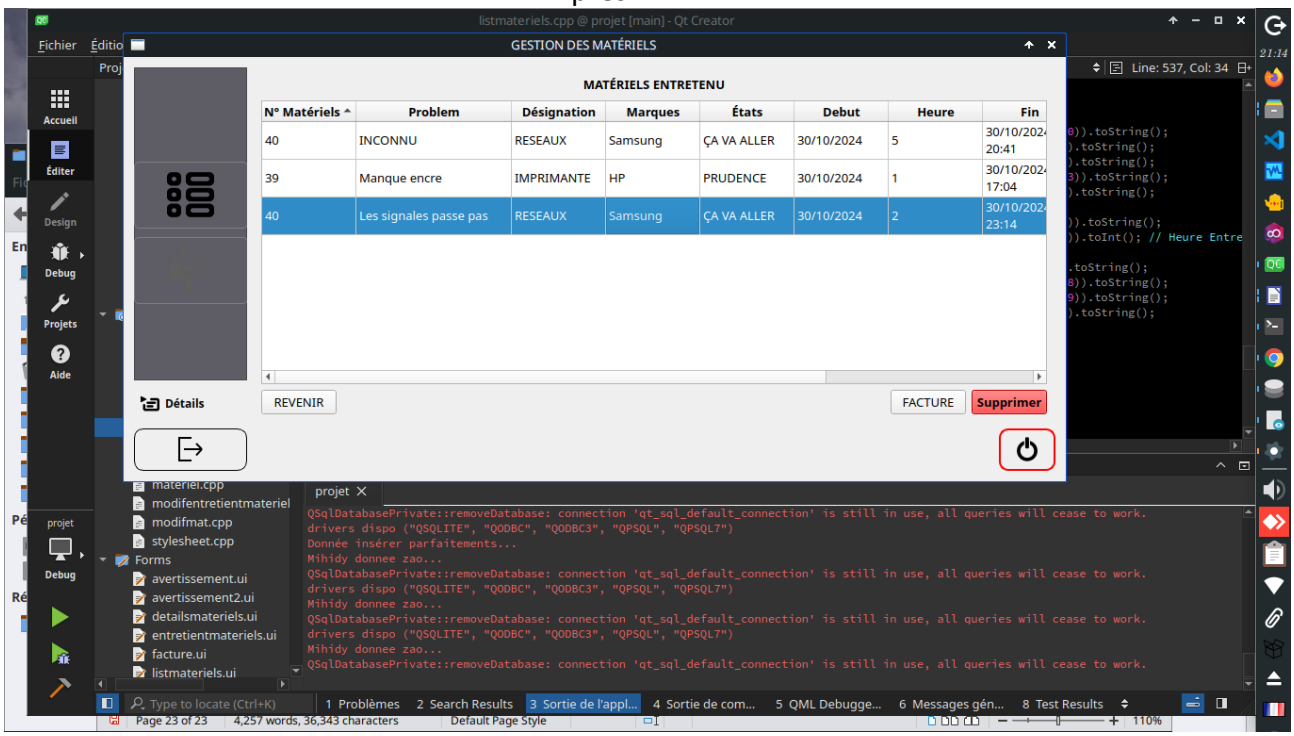

Fig. 11

- *Fenêtre facture*

Dans la fenêtre Entretient, l'Utilisateur peut gérer l'Entretient en cliquant sur une des données, en double cliquant pour modifier, après ça elle peut voir et éditer en PDF la facture de cette entretient

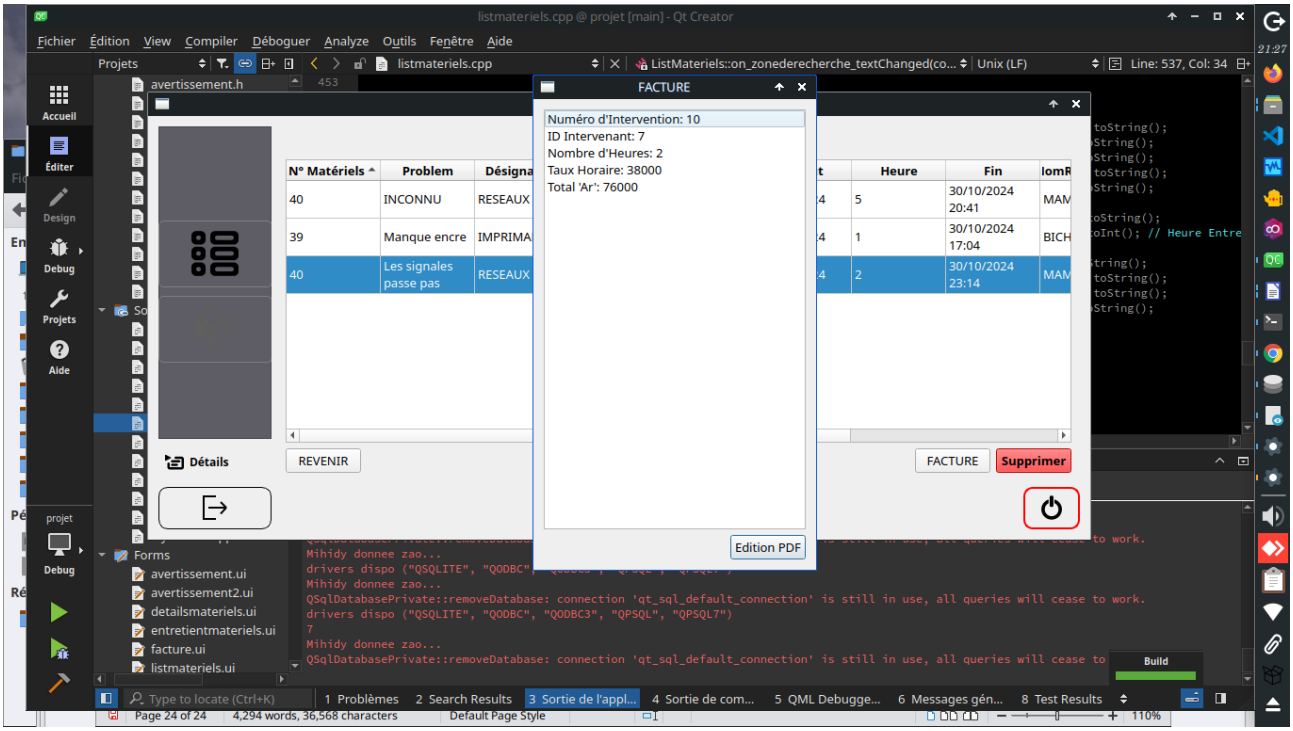

Fig. 12

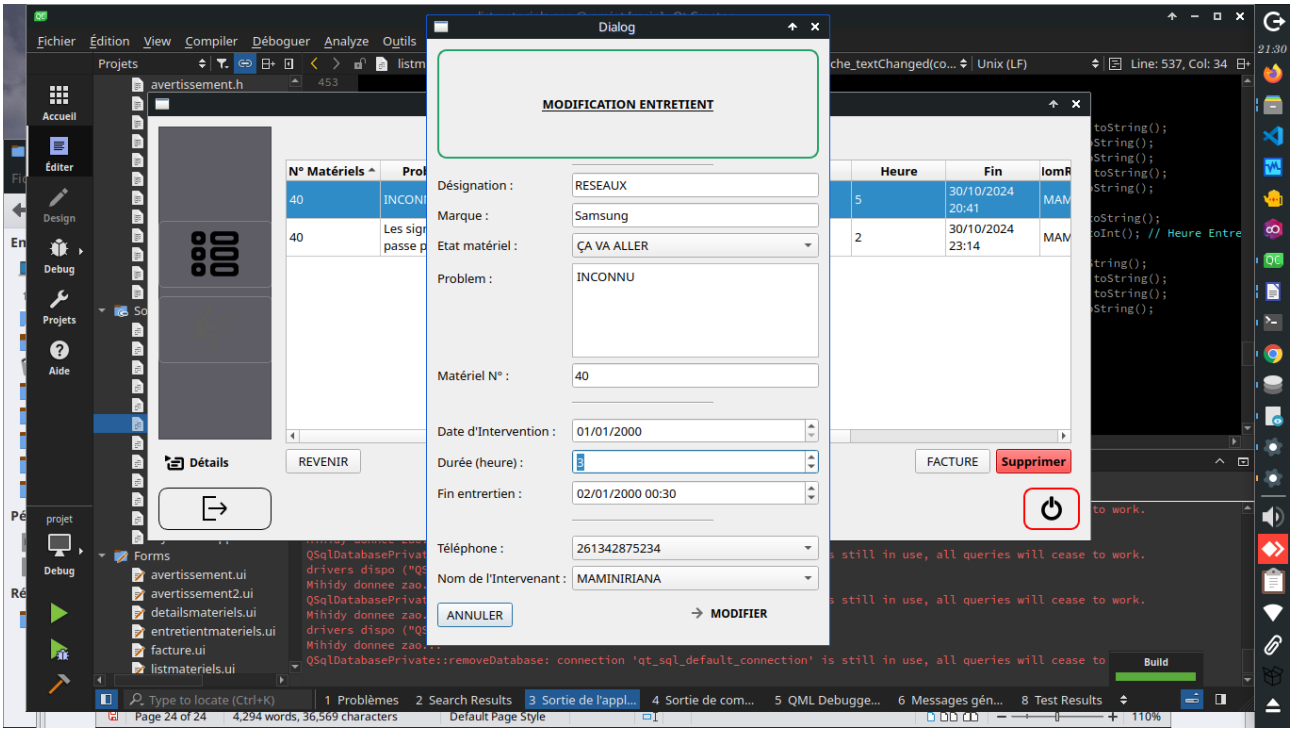

Fig. 13

- *Changement Utilisateur*

Enfin, après avoir fini de modifier quelque matériels, l'Utilisateur peut se déconnecter en laissant libre place aux autres Utilisateurs

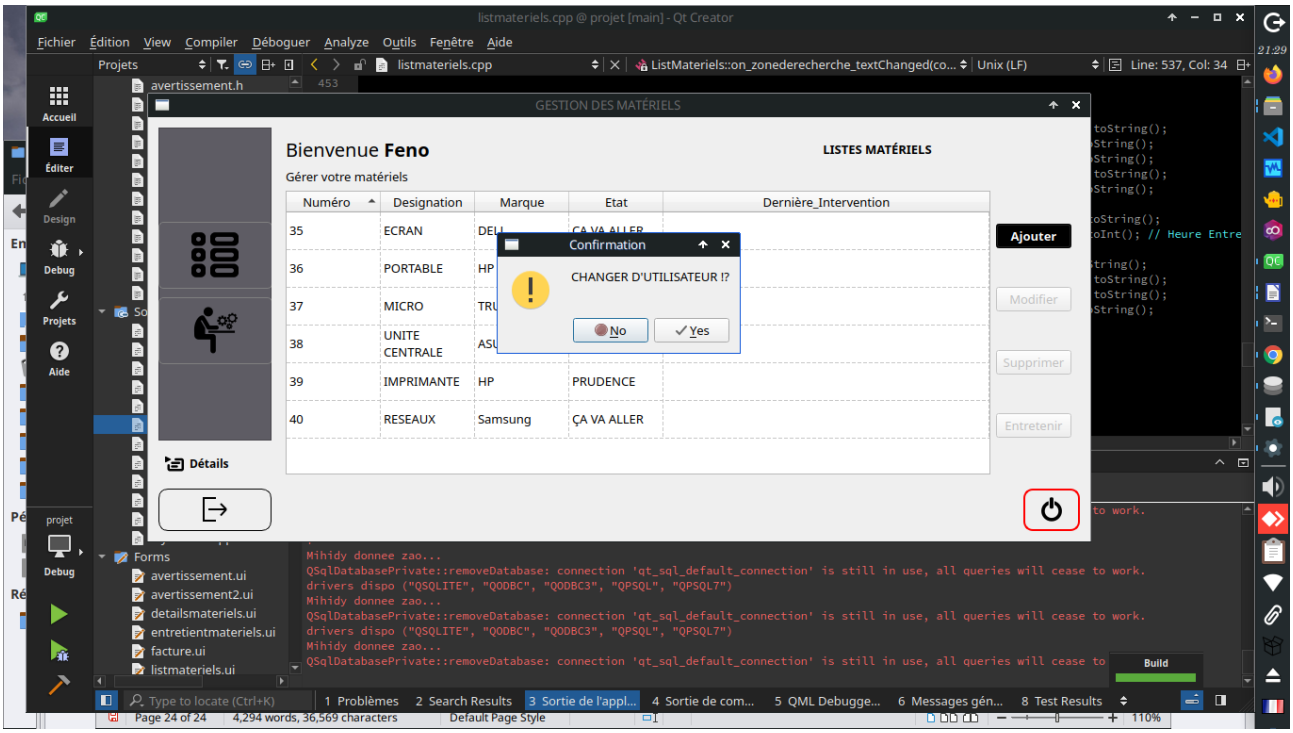

Fig. 14

Conclusion

Dans un monde où la rapidité de l'évolution technologique défie constamment les entreprises dans la gestion de leurs matériels informatiques, la nécessité d'une solution efficace et adaptable devient impérative. Notre exploration des défis complexes associés à la gestion manuelle des stocks et des opportunités offertes par une approche technologique a jeté les bases d'une réflexion approfondie.

À la lumière de ces considérations, il devient évident que la création d'une application dédiée, alignée sur une base de données appropriée, représente une réponse prometteuse. L'ensemble du processus, depuis la conception jusqu'à l'identification des fonctionnalités clés, s'inscrit dans une démarche visant à simplifier et optimiser la gestion des matériels informatiques au sein des organisations.

Notre exploration nous a guidés à travers les rouages de cette quête pour une solution efficace. Les enjeux spécifiques ont été dévoilés, les choix technologiques et les principes directeurs ont été mis en lumière, et les fonctionnalités clés ont été identifiées. Ainsi, une application bien conçue, adaptée aux besoins évolutifs des entreprises, promet non seulement de faciliter la gestion quotidienne des matériels informatiques, mais également d'assurer fiabilité et constance dans un environnement en perpétuelle évolution technologique. Ce périple conceptuel laisse entrevoir un horizon où la gestion des matériels informatiques devient une tâche harmonieuse et efficiente, grâce à une technologie pensée pour répondre aux défis du présent et anticiper les exigences du futur.

Bibliographie

RAKOTOMAMINIRIANA Fanomezaniavo F S, NOMENJANAHARY Martela B, MBOLA Tsimahory T, ANDRIANIAINA Florin B, RAZAFIMANANDROVOLA Jonathan, Conception et réalisation d’une application pour la gestion de matériels informatique. Toliara :R Jonathan ,2024.54p

Webographie

Conception et Réalisation : http://wwwenitoliara ,consulté le 30 Octobre 2024

Table des matières

Résumé et Abstract

La réalisation de ce projet s'est faite en plusieurs étapes, incluant l'analyse des besoins des utilisateurs, la conception de l'interface, le développement et les tests. En intégrant des technologies modernes, notre application garantit non seulement une gestion optimale du matériel, mais aussi une expérience utilisateur fluide et efficace.

The IT hardware market shares the same challenges as other major sectors when it comes to inventory management. This report presents an alternative solution through our dedicated hardware management application. Our approach is based on two fundamental pillars: the use of a Relational Database Management System (RDBMS), implemented with MySQL, and the integration of the Qt Framework combined with C++ programming.

MySQL has proven to be a judicious choice for the creation of a database adapted to the management of computer hardware, offering simplicity and efficiency thanks to its SQL language. The latter enables consistent data manipulation, from creation to deletion. For its part, the Qt Framework stands out as a powerful tool for designing user-friendly graphic interfaces, while C++ programming reinforces the application's robustness.

The current application provides an effective response to the problem of managing IT hardware inventories. However, as in other domains, it has its limitations, both visual and technical. These limitations are not insurmountable obstacles, but rather opportunities for improvement. The addition of further functionalities or the integration of more advanced technologies can enable the application to remain at the cutting edge and to adapt to changing needs.