



**Faculté des Sciences et Technique de Tanger
Département Informatique
Filière Génie Informatique
Cycle Licence**

Numéro d'ordre Étudiant : 1800000025

Application d'un jeu Apprentissage d'Anglais

Projet tutoré et présenté par :
VEZOLO MBETTE Asseht Rosaire

Sous la direction de :
Madame Sanae KHALI ISSA

Soutenu le : ...//...//....

Présenté Devant la Responsable :

Professeur :**Sanae KHALI ISSA**

Année Universitaire : 2023-2024

Table des matières

1		3
1.1	Préparation projet	3
1.2	Organisation du Projet	3
1.2.1	Encadrant du projet	3
1.2.2	Planning du Diagramme de Gant :	3
1.3	Langage Utilisé	4
1.4	Modélisation UML	5
1.4.1	Diagramme de classe :	5
1.4.2	Diagramme de use case :	6
1.4.3	Diagramme d'activité :	6
1.4.4	Diagramme séquence :	7
1.5	Création des Niveaux et prise de mains	8
1.5.1	Fonction et syntaxe niveau et Classe intégré	8
1.6	Interface jeu et Niveau	17
1.7	Difficulté Rencontrer	20
1.8	Profit Tirer	20
1.9	Conclusion	20

Chapitre 1

1.1 Préparation projet

Le projet est la mise en place d'une application permettant d'apprendre les cours d'anglais. Ceci permet de mettre en évidence une série des question QCM sur la grammaire action de coordination tout sur texte a trou.

1.2 Organisation du Projet

Membre du projet	responsabilité
VEZOLO MBETTE Asseht Rosaire	Ensemble du projet

1.2.1 Encadrant du projet

Professeur	responsabilité
Sanae KHALI ISSA	Évaluation et Notation projet

1.2.2 Planning du Diagramme de Gant :



1.3 Langage Utilisé

Dans ce projet j'ai utilisé des logiciels suivant :

PowerAMC : pour la mise en place de la modélisation UML



Latex & Word : est un langage de balise pour la mise en place et la saisie du rapport.



QtC++ : est un Framework ide c++ qui permet de développer les application desktop.



Qt Design Studio : est une interface de Design utiliser par Qt (python, c++ etc..) pour développer l'interface utilisateur.



Json Package : pour la mise en place d'une data base local

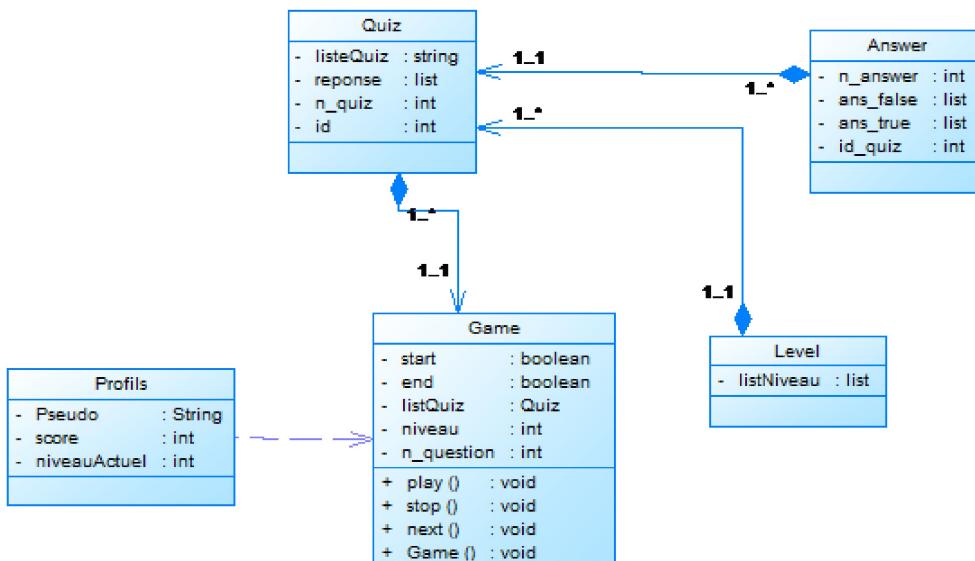


1.4 Modélisation UML

1.4.1 Diagramme de classe :

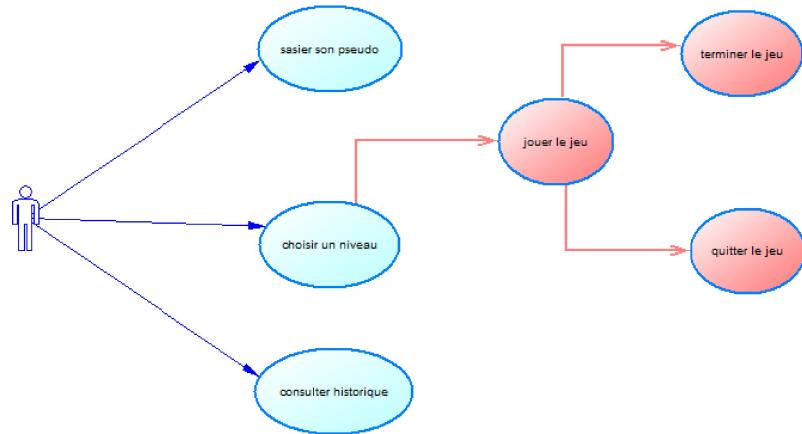
Le diagramme de classes est utilisé pour représenter la structure d'un système en spécifiant les différentes classes d'objets ainsi que les relations entre ces classes.

Dans notre système d'application, le diagramme de classes est utilisé pour représenter les différentes entités du système, telles que les Quiz, les Answers, etc., ainsi que les relations entre ces entités.



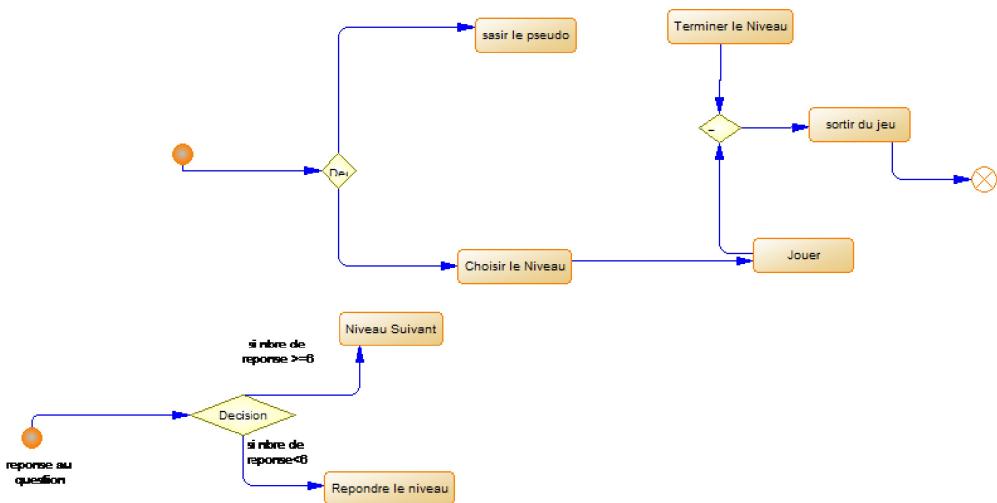
1.4.2 Diagramme de use case :

Ce diagramme nous permet de spécifier le rôle de l'utilisateur avec le système ou l'application

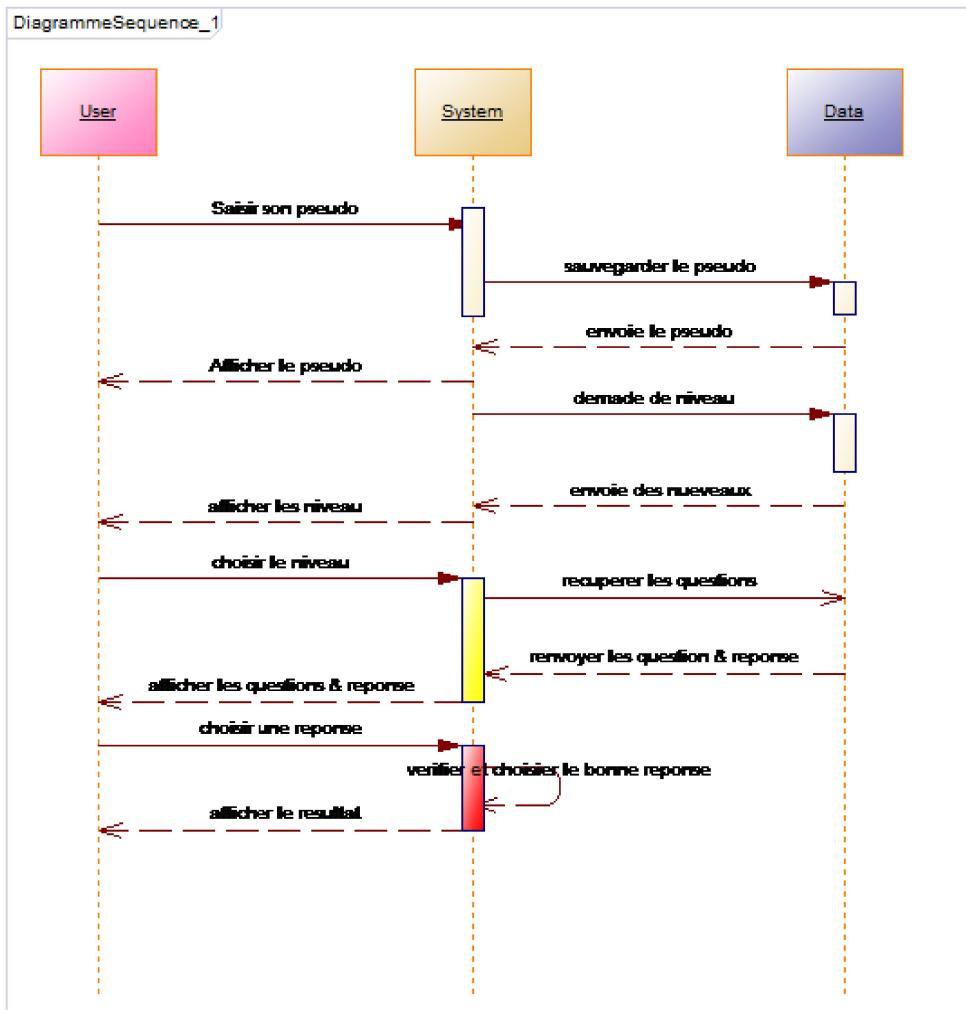


1.4.3 Diagramme d'activité :

Les diagrammes d'activités sont utilisés pour représenter les processus métiers d'un système, en spécifiant les différentes étapes ainsi que les conditions de transition entre ces étapes. Dans notre système d'application, les diagrammes d'activités sont utilisés pour modéliser les processus de connexion, de saisie de chemin et de jouer le jeu.



1.4.4 Diagramme séquence :



1.5 Creation des Niveaux et prise de mains

1.5.1 Fonction et syntaxe niveau et Classe integre

Avant de faire la saisie du code, il fallait faire quelque modification au niveau du code source donnee par defaut lors de la creation du nouveau projet sur **Qt**. La 1er choses la mise en place de l’interface du jeu a travers le fichier **Ui** suivant :

```
<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>LearnEnglish</class>

<widget class="QMainWindow" name="LearnEnglish">

<property name="geometry">

<rect>
<x>0</x>
<y>0</y>
<width>539</width>
<height>336</height>
</rect>
```

La resolution qui sera utiliser pour la lecran sera **539X336 px**. La configuration de cette interface nous a permis aussi de faire la mise en place des classes lier a chaque evenement. Chaque classe **.cpp** ou **.h** joue un role specifique.

Les classes **.h** nous permet de declarer des attributs, des methodes, et les appels include par contre, les classes **.cpp** permet d’appeler les methodes et les attributs disponible dans les fichiers **.h** pour la creation a la manipulation des objets.

Les classes .h sont les suivantes :

User.h : Contient le nom utilisateur, l'historique, l'affiche score permettant de récupérer les données saisies sur l'interface avec le méthode **QString** et **QSittings** (des méthode propre de Qt).

```

#ifndef USER_H
#define USER_H
#include<QStringList>
#include <QString>
class user{
private:
    QString pseudo;
    int score;
    int niveauActuel;
    QStringList historique;
public:
    user();
    // appeler setter
    void setPseudo(QString pseudo){
        this->pseudo=pseudo;
    }
    void setScore(int score){
        this->score=score;
    }
    void setNiveauActuel(int niveauActuel){
        this->niveauActuel=niveauActuel;
    }
    QString getPseudo(void){
        return(this->pseudo);
    }
    int getScore(void){
        return(this->score);
    }
    int getNiveauActuel(void){
        return(this->niveauActuel);
    }
    QStringList getHistorique(void){
        return(this->historique);
    }
};
#endif // USER_H

```

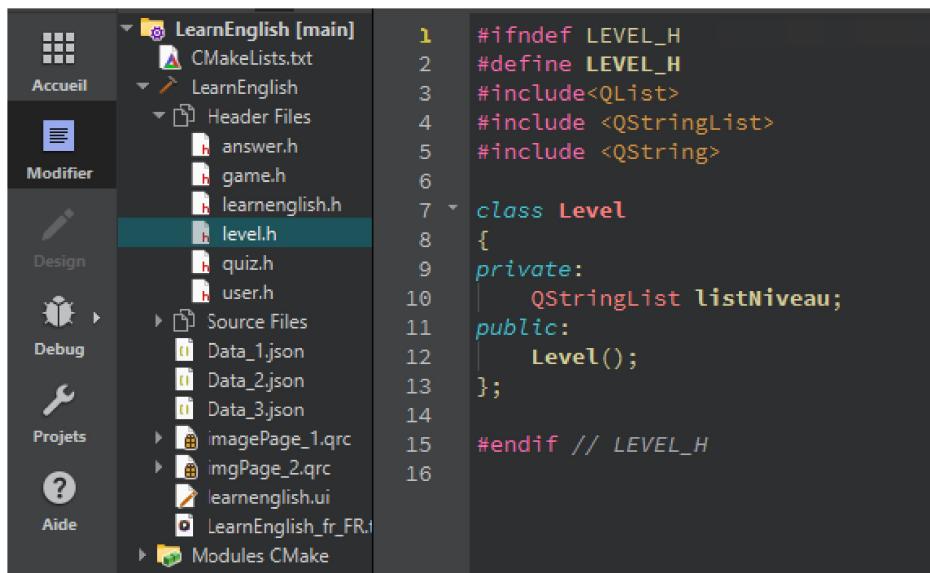
Level.h : Contient la liste de niveau et le niveau étape par étape.

```

#ifndef LEVEL_H
#define LEVEL_H
#include<QList>
#include <QStringList>
#include <QString>
class Level
{
private:
    QStringList listNiveau;
public:
    Level();
};
#endif // LEVEL_H

```

LearnEnglish.h : Contient les attributs et méthode d'appel depuis l'interface ui pour l'appel des boutons lors des cliques ils sont déclarés dans **private slots**, la fonction d'appel sur menu bar, il contient aussi le destructeur.



```

LearnEnglish [main]
  CMakeLists.txt
  LearnEnglish
    Header Files
      answer.h
      game.h
      learnenglish.h
      level.h // Selected file
      quiz.h
      user.h
    Source Files
      Data_1.json
      Data_2.json
      Data_3.json
    imagePage_1.qrc
    imgPage_2.qrc
    learnenglish.ui
    LearnEnglish_fr_FR.i
  Modules CMake

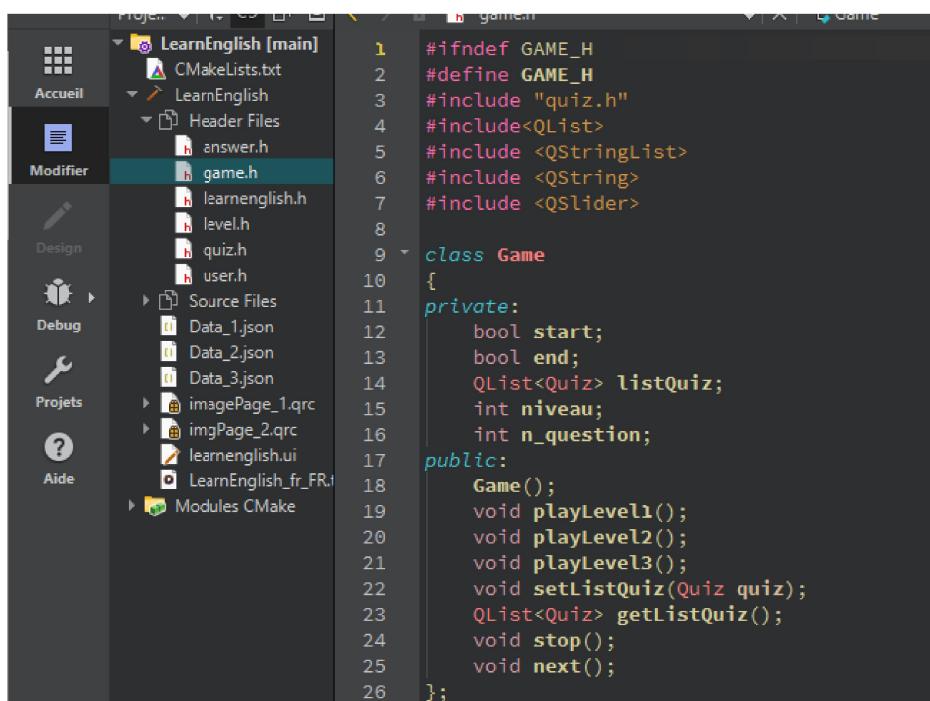
```

```

1 #ifndef LEVEL_H
2 #define LEVEL_H
3 #include<QList>
4 #include &ltQStringList>
5 #include &ltQString>
6
7 class Level
8 {
9 private:
10   QStringList listNiveau;
11 public:
12   Level();
13 };
14
15 #endif // LEVEL_H
16

```

Game.h : Contient les méthodes et les fonctions pour la mise en place du jeu.



```

LearnEnglish [main]
  CMakeLists.txt
  LearnEnglish
    Header Files
      answer.h
      game.h // Selected file
      learnenglish.h
      level.h
      quiz.h
      user.h
    Source Files
      Data_1.json
      Data_2.json
      Data_3.json
    imagePage_1.qrc
    imgPage_2.qrc
    learnenglish.ui
    LearnEnglish_fr_FR.i
  Modules CMake

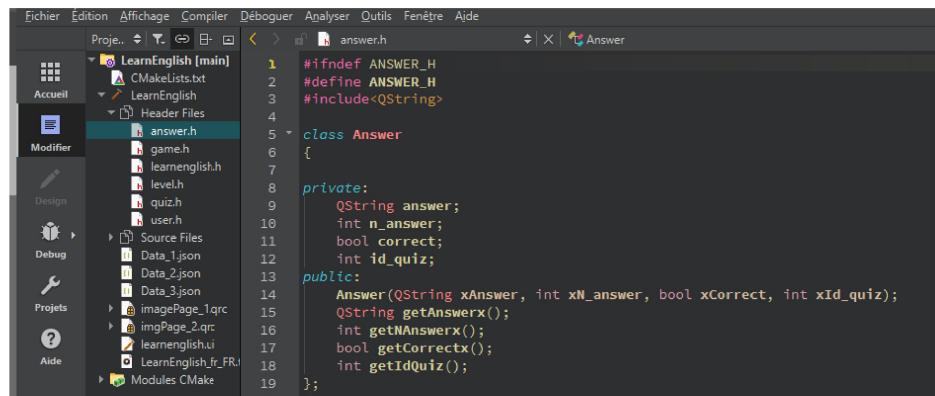
```

```

1 #ifndef GAME_H
2 #define GAME_H
3 #include "quiz.h"
4 #include<QList>
5 #include &ltQStringList>
6 #include &ltQString>
7 #include <QSlider>
8
9 class Game
10 {
11 private:
12   bool start;
13   bool end;
14   QList<Quiz> listQuiz;
15   int niveau;
16   int n_question;
17 public:
18   Game();
19   void playLevel1();
20   void playLevel2();
21   void playLevel3();
22   void setListQuiz(Quiz quiz);
23   QList<Quiz> getListQuiz();
24   void stop();
25   void next();
26 };

```

Answer.h : Contient les méthodes et les fonctions pour la mise en place du jeu.

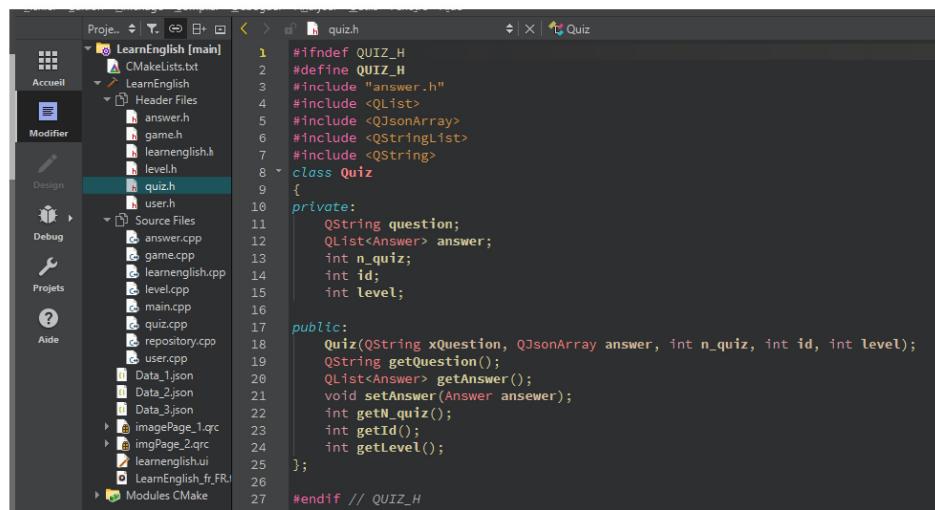


```
#ifndef ANSWER_H
#define ANSWER_H
#include <QString>

class Answer
{
private:
    QString answer;
    int n_answer;
    bool correct;
    int id_quiz;
public:
    Answer(QString xAnswer, int xN_answer, bool xCorrect, int xId_quiz);
    QString getAnswer();
    int getNAnswer();
    bool getCorrect();
    int getIdQuiz();
};

#endif // ANSWER_H
```

Quiz.h : Contient des attributs et méthode qui Nous permet de récupérer la liste des questions depuis le fichier json du niveau concerné



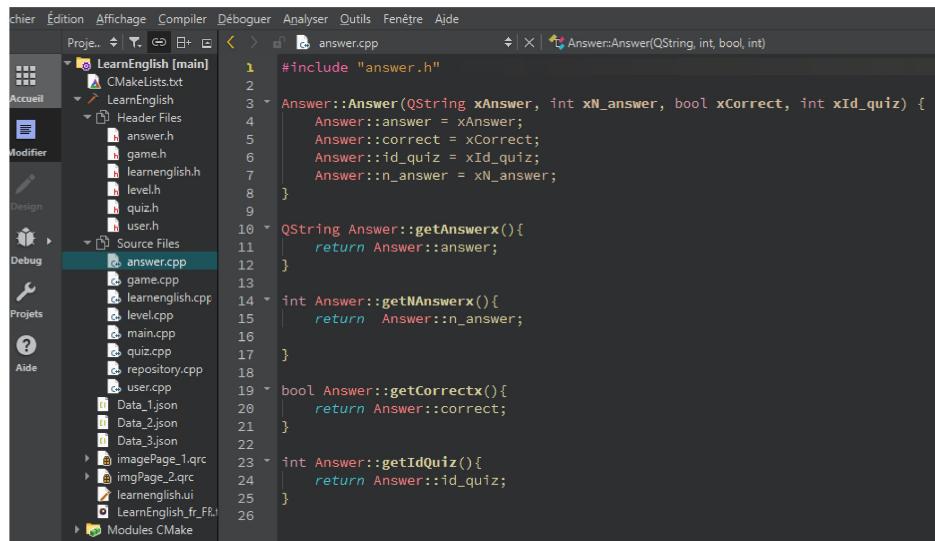
```
#ifndef QUIZ_H
#define QUIZ_H
#include "answer.h"
#include <QList>
#include <QJsonArray>
#include <QStringList>
#include <QString>

class Quiz
{
private:
    QString question;
    QList<Answer> answer;
    int n_quiz;
    int id;
    int level;
public:
    Quiz(QString xQuestion, QJsonArray answer, int n_quiz, int id, int level);
    QString getQuestion();
    QList<Answer> getAnswer();
    void setAnswer(Answer ansewer);
    int getN_quiz();
    int getId();
    int getLevel();
};

#endif // QUIZ_H
```

Les classes .cpp sont les suivantes :

Answer.cpp : Nous permettons de récupérer les attributs et méthode déclarer dans le fichier [answer.h](#)



```
#include "answer.h"

Answer::Answer(QString xAnswer, int xN_answer, bool xCorrect, int xId_quiz) {
    Answer::answer = xAnswer;
    Answer::correct = xCorrect;
    Answer::id_quiz = xId_quiz;
    Answer::n_answer = xN_answer;
}

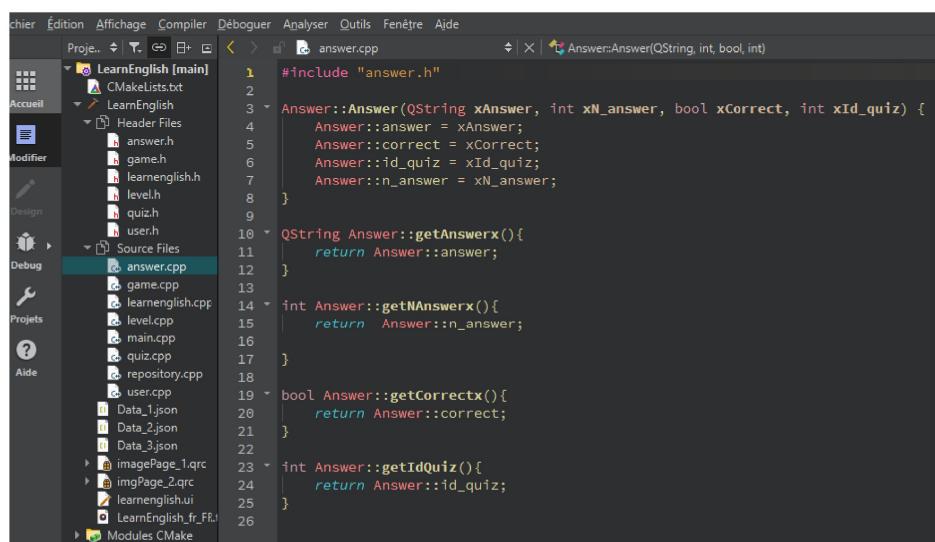
QString Answer::getAnswerx(){
    return Answer::answer;
}

int Answer::getNAnswerx(){
    return Answer::n_answer;
}

bool Answer::getCorrectx(){
    return Answer::correct;
}

int Answer::getIdQuiz(){
    return Answer::id_quiz;
}
```

Game.cpp : Il nous permet de récupérer, les fichiers [.h](#) de [quiz](#) et [game](#) permettons la mise en place des niveau [Play Game](#) il permet aussi de la récupération des données depuis le fichier [json](#) de chaque niveau.



```
#include "answer.h"

Answer::Answer(QString xAnswer, int xN_answer, bool xCorrect, int xId_quiz) {
    Answer::answer = xAnswer;
    Answer::correct = xCorrect;
    Answer::id_quiz = xId_quiz;
    Answer::n_answer = xN_answer;
}

QString Answer::getAnswerx(){
    return Answer::answer;
}

int Answer::getNAnswerx(){
    return Answer::n_answer;
}

bool Answer::getCorrectx(){
    return Answer::correct;
}

int Answer::getIdQuiz(){
    return Answer::id_quiz;
}
```

LearnEnglish.cpp : Il s'occupe de la grande partie du projet en récupérerons l'interface ui, les manipulations des objets, le nom d'user, les attributs et méthode déclarer dans LearnEnglish.h, le système d'entré sortie, sur le clavier.

```

1 #include "learnenglish.h"
2 #include "./ui_learnenglish.h"
3 #include "QSettings"
4 #include "QDebug"
5 #include "QLineEdit"
6 #include "game.h"
7 #include <QMessageBox>
8 #include <QListView>
9 #include <QStandardItemModel>
10 #include <QRadioButton>
11
12 LearnEnglish::LearnEnglish(QWidget *parent)
13 : QMainWindow(parent),
14   , ui(new Ui::LearnEnglish)
15 {
16   ui->setupUi(this);
17   QSettings settings("LearnEnglish", "InterfaceSouscription");
18   QString user = settings.value("pseudo").toString(); // recuper le pseudo
19   //settings.remove("pseudo");
20   if(user.isEmpty()){
21     ui->stackedWidget->setcurrentIndex(ui->stackedWidget->currentIndex() + 1); //afficher la page de 1 pseudo
22   }else{
23     ui->label_8->setText(user); // recuperé l'element dans le label
24     ui->stackedWidget->setcurrentIndex(0); //afficher la 2eme pages
25   }
26   connect(ui->pushButton_6, &QPushButton::clicked, this, &QWidget::close);
27   connect(ui->actionA_Propos, &QAction::triggered, this, &LearnEnglish::a_propos); // fenetre a propos
28
29 }
30
31

```

```

31
32 LearnEnglish::~LearnEnglish()
33 {
34   delete ui;
35 }
36 void LearnEnglish::a_propos(void){
37   QMessageBox::information(this, "A Propos ", "ceci est un message est une application pour tester votre niveau en anglais");
38 }
39
40 void LearnEnglish::on_pushButton_clicked()
41 {
42   try { //enregister le pseudo
43     QString pseudo = ui->field_pseudo->text(); // saisir la valeur dans le champ
44     QSettings settings("LearnEnglish", "InterfaceSouscription");
45     settings.setValue("pseudo", pseudo);
46     // qDebug<<"la votre pseudo est: "<<pseudo;
47     ui->label_8->setText(settings.value("pseudo").toString()); // recuperé l'element dans le label
48     ui->stackedWidget->setcurrentIndex(0); //afficher la page suivante
49   } catch (...) {
50   }
51 }
52
53
54
55

```

```

56
57 void LearnEnglish::on_pushButton_5_clicked()
58 {
59   ui->stackedWidget->setcurrentIndex(5);
60 }
61
62
63 void LearnEnglish::on_pushButton_2_clicked()
64 {
65   QSettings settings("LearnEnglish", "InterfaceSouscription");
66   //Index de la question a affiché
67   int questionId = 1;
68   settings.setValue("questionId", questionId);
69   // Créer un modèle de données
70   QStandardItemModel *model = new QStandardItemModel;
71   Game game = Game();
72
73   //Initialisation des questions
74   game.playLevel1();
75
76   //La liste des questions
77   QList<Quiz> questions = game.getListQuiz();
78
79   //Recherche de la question via son id
80   foreach (Quiz quiz, questions) {
81     //Vérification de l'id de la question
82     if(quiz.getId() == questionId){
83       //Quiz quiz = questions.first();
84       //Le nombre de question
85       int nbre = questions.length();
86       QString nbreQuestion = QString::number(questionId)+"+"+QString::number(nbre);
87
88
89
90
91
92
93
94
95
96
97

```

```

ui->label_11->setText(nbreQuestion);
//qDebug() <<"Valeur: "<< question.length();
ui->label_9->setText(quiz.getQuestion());
foreach (auto valeur, quiz.getAnswer()) {
    QStandardItem *item = new QStandardItem(valeur.getAnswerx());
    bool correct = valeur.getCorrectx();
    if(correct){
        settings.setValue("answerCorrect", valeur.getAnswerx());
    }
    QRadioButton *boutonRadio = new QRadioButton();
    item->setData(QVariant::fromValue(boutonRadio), Qt::DecorationRole);
    model->appendRow(item);
}
//qDebug() <<"Valeur: "<< question.first().getQuestion();
QListView *listView = ui->listView;
listView->setModel(model);
ui->stackedWidget->setCurrentIndex(1); //niveau 1
break;
}

```

```

void LearnEnglish::on_pushButton_3_clicked()
{
    QSettings settings("LeranEnglish", "InterfaceSouscription");
    //Index de la question à affiché
    int questionId = 1;
    settings.setValue("questionId", questionId);
    //Créer un modèle de données
    QStandardItemModel *model = new QStandardItemModel;
    Game game = Game();
    //Initialisation des questions
    game.playLevel2();
    //La liste des questions
    QList<Quiz> questions = game.getListQuiz();
    //qDebug() <<"Valeur: "<< questions.length();
    //Recherche de la question via son id
    foreach (Quiz quiz, questions) {
        //Vérification de l'id de la question
        if(quiz.getId() == questionId){
            //Quiz quiz = questions.first();
            //Le nombre de question
            int nbre = questions.length();
            QString nbreQuestion = QString::number(questionId)+"/"+QString::number(nbre);

            ui->label_28->setText(nbreQuestion);

            ui->label_45->setText(quiz.getQuestion());
            foreach (auto valeur, quiz.getAnswer()) {
                QStandardItem *item = new QStandardItem(valeur.getAnswerx());
                bool correct = valeur.getCorrectx();
                if(correct){
                    settings.setValue("answerCorrect", valeur.getAnswerx());
                }
                QRadioButton *boutonRadio = new QRadioButton();
                item->setData(QVariant::fromValue(boutonRadio), Qt::DecorationRole);
                model->appendRow(item);
            }
            //qDebug() <<"Valeur: "<< quiz.getQuestion();
            QListView *listView = ui->listView_3;
            listView->setModel(model);
            ui->stackedWidget->setCurrentIndex(2); //niveau 2
            break;
        }
    }
}

```

```

void LearnEnglish::on_pushButton_4_clicked()
{
    QSettings settings("LeranEnglish", "InterfaceSouscription");
    //Index de la question à affiché
    int questionId = 1;
    settings.setValue("questionId", questionId);
    //Créer un modèle de données
    QStandardItemModel *model = new QStandardItemModel;
    Game game = Game();
    //Initialisation des questions
    game.playLevel3();
    //La liste des questions
    QList<Quiz> questions = game.getListQuiz();
    //qDebug() <<"Valeur: "<< questions.length();
    //Recherche de la question via son id
    foreach (Quiz quiz, questions) {
        //Vérification de l'id de la question
        if(quiz.getId() == questionId){
            //Quiz quiz = questions.first();
            //Le nombre de question
            int nbre = questions.length();
            QString nbreQuestion = QString::number(questionId)+"/"+QString::number(nbre);

            ui->label_50->setText(nbreQuestion);
        }
    }
}

```

```

195 ui->label_47->setText(quiz.getQuestion());
196 foreach (auto valeur, quiz.getAnswer()) {
197     QStandardItem *item = new QStandardItem(valeur.getAnswerx());
198     bool correct = valeur.getCorrectx();
199     if(correct){
200         settings.setValue("answerCorrect", valeur.getAnswerx());
201     }
202     QRadioButton *boutonRadio = new QRadioButton();
203     item->setData(QVariant::fromValue(boutonRadio), Qt::DecorationRole);
204     model->appendRow(item);
205 }
206 //qDebug() <<"Valeur: "<< quiz.getQuestion();
207 QListWidget *listView = ui->listView_6;
208 listView->setModel(model);
209 ui->stackedWidget->setCurrentIndex(3); //niveau 3
210 break;
211 }

```

Main.cpp : Permet de faire appeler de l'écran de démarrage, et la compilation sur l'ensemble des classes, de l'ouverture et fermeture de l'interface utilisateur.

```

1 #include "LearnEnglish.h"
2
3 #include <QApplication>
4 #include <QLocale>
5 #include <QTranslator>
6 #include <QSplashScreen>
7 #include <QTimer>
8 #include "LearnEnglish.h"
9
10 int main(int argc, char *argv[])
11 {
12     QApplication a(argc, argv);
13
14     QTranslator translator;
15     const QStringList uilanguages = QLocale::system().uiLanguages();
16     for (const QString &locale : uilanguages) {
17         const QString baseName = "LearnEnglish_" + QLocale(locale).name();
18         if (translator.load(":/i18n/" + baseName)) {
19             a.installTranslator(&translator);
20             break;
21         }
22     }
23     LearnEnglish w;
24     // afficher une image de demarage
25     QSplashScreen *splash = new QSplashScreen;
26     //QPixmap pixmap("Desktop\project-c++\Game_learn\game_learn\Ressource\img_demarrege");
27     //splash->setPixmap(QPixmap("C:/Users/Rosaire-Verzolo/Desktop/projet-c++/game_learn/Ressource/img_demarrege/img.png"));
28     splash->show();
29
30     //ferre la fermeture
31     QTimer::singleShot(3000,splash, SLOT(close()));
32     //creer la fenetre principale
33     QTimer::singleShot(3000, &w, SLOT(show()));
34
35
36     return a.exec();
37 }
38

```

Quiz.cpp : Récupère et initialise les attributs et classe déclarer dans quiz.h

```

1 #include "quiz.h"
2 #include "answer.h"
3
4 Quiz::Quiz(QString xquestion, QJsonArray xAnswer, int xm_quiz, int xId, int xlevel) {
5     Quiz::id = xId;
6     Quiz::question = xQuestion;
7     Quiz::level = xLevel;
8
9     for(const QVariant &item: xAnswer) {
10         Quiz::setAnswer(Answer(item["rep"].toString(), item["correct"].toBool(), xId));
11     }
12 }
13
14 int Quiz::getLevel(){
15     return Quiz::level;
16 }
17
18 QList<Answer> Quiz::getAnswer() {
19     return answer;
20 }
21
22 void Quiz::setAnswer(Answer xAnswer){
23     Quiz::answer.append(xAnswer);
24 }
25
26
27 int Quiz::getId(){
28     return Quiz::id;
29 }
30
31 int Quiz::getQuiz(){
32     return Quiz::m_quiz;
33 }
34
35 QString Quiz::getQuestion(){
36     return Quiz::question;
37 }

```

Repository.cpp : Il nous permet de récupérer les fichier json avec les fonctions propres a Qt comme : `QJsonObject`, `QJsonDocument` ect... son rôle ultime la manipulation des données json.

```

1 //include <QList>
2 //include <QCoreApplication>
3 //include <QFile>
4 //include <QJsonDocument>
5 //include <QJsonObject>
6 //include <QJsonValue>
7 //include <QJsonArray>
8 //include <QDebug>
9
10 * class Repository
11 {
12     Q_OBJECT
13     private:
14
15 public:
16     Repository();
17
18     static QJsonObject questionLevel1()
19     {
20         // Nom du fichier JSON
21         QString nomFichier = "C:/Users/Rosarie-Vezolo/Desktop/projet-c++/Game_learn/Game_learn/Data_1.json";
22         // Ouvrir le fichier en mode lecture
23         QFile file(nomFichier);
24         if (!file.open(QIODevice::ReadOnly | QIODevice::Text)) {
25             qDebug() << "Impossible d'ouvrir le fichier JSON.";
26         }
27         return quiz;
28     }

```

```

29     // Lire les données du fichier
30     QByteArray donneesJson = file.readAll();
31
32     // Fermer le fichier
33     file.close();
34
35     // Convertir les données JSON en document JSON
36     QJsonDocument document = QJsonDocument::fromJson(donneesJson);
37
38     // Vérifier si l'analyse JSON a réussi
39     if (document.isNull()) {
40         qDebug() << "Erreur d'analyse JSON.";
41         return quiz;
42     }
43
44     // Extraire l'objet JSON racine
45     quiz = document.object();
46
47     return quiz;
48 }
49
50 static QJsonObject questionLevel2()
51 {
52     // Nom du fichier JSON
53     QString nomFichier = "C:/Users/Rosarie-Vezolo/Desktop/projet-c++/Game_learn/Game_learn/Data_2.json";
54     // Ouvrir le fichier en mode lecture
55     QFile file(nomFichier);
56     if (!file.open(QIODevice::ReadOnly | QIODevice::Text)) {
57         qDebug() << "Impossible d'ouvrir le fichier JSON.";
58     }
59     return quiz;
60 }
61
62

```

```

63     // Fermer le fichier
64     file.close();
65
66     // Convertir les données JSON en document JSON
67     QJsonDocument document = QJsonDocument::fromJson(donneesJson);
68
69     // Vérifier si l'analyse JSON a réussi
70     if (document.isNull()) {
71         qDebug() << "Erreur d'analyse JSON.";
72         return quiz;
73     }
74
75     // Extraire l'objet JSON racine
76     quiz = document.object();
77
78     return quiz;
79 }
80
81 static QJsonObject questionLevel3()
82 {
83     // Nom du fichier JSON
84     QString nomFichier = "C:/Users/Rosarie-Vezolo/Desktop/projet-c++/Game_learn/Game_learn/Data_3.json";
85     // Ouvrir le fichier en mode lecture
86     QFile file(nomFichier);
87     if (!file.open(QIODevice::ReadOnly | QIODevice::Text)) {
88         qDebug() << "Impossible d'ouvrir le fichier JSON.";
89     }
90     return quiz;
91 }
92
93
94     // Lire les données du fichier
95     QByteArray donneesJson = file.readAll();
96
97

```

```

answer.cpp      92     qDebug() << "Impossible d'ouvrir le fichier JSON." ;
game.cpp       93     }
levelenglish.cpp 94     return quiz;
main.cpp        95     // Lire les données du fichier
                    QByteArray donneesJson = file.readAll();
repository.cpp  96     // Fermer le fichier
                    file.close();
                    // Convertir les données JSON en document JSON
                    QJsonDocument document = QJsonDocument::fromJson(donneesJson);
                    // Vérifier si l'analyse JSON a réussi
                    if (document.isNull()) {
                        qDebug() << "Erreur d'analyse JSON." ;
                        return quiz;
                    }
                    // Extraire l'objet JSON racine
                    quiz = document.object();
}
repository.cpp  111
repository.cpp  112
repository.cpp  113
repository.cpp  114
repository.cpp  115
repository.cpp  116
repository.cpp  117

```

Fichier Json :

Les fichiers nous permettent d'avoir une base de donnée local pour la mise en place des question et réponse niveau 1, 2, 3.

Je tiens à préciser pour l'interface **Qt** pour la récupération des données comme **le Pseudo** et **le niveau** actuel, on utilise **QString** et **Qsettings** :

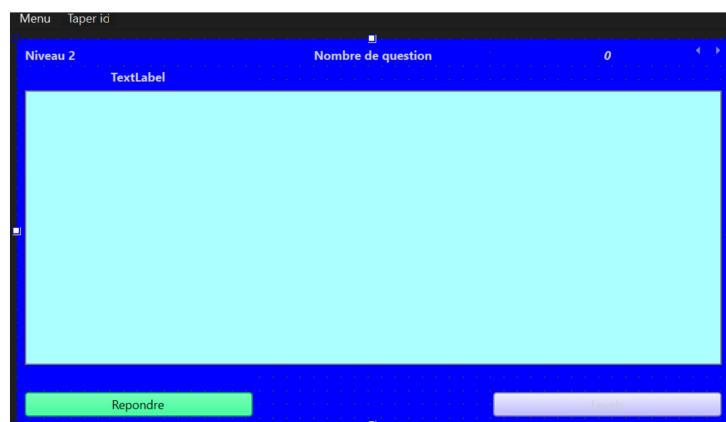
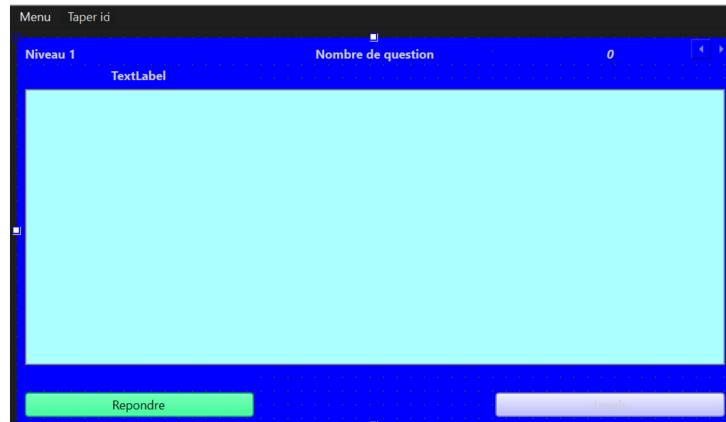
```

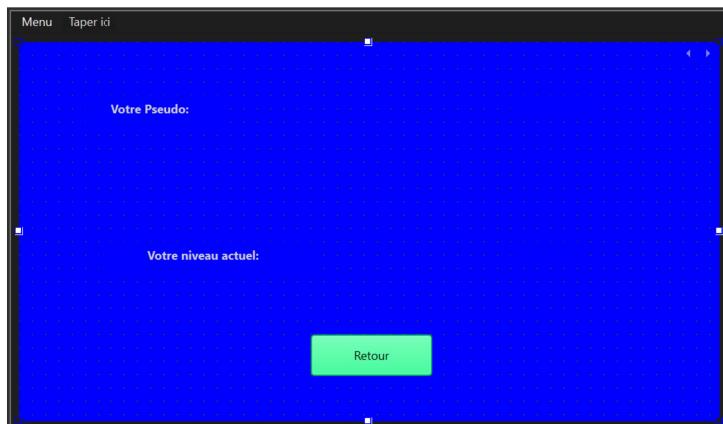
QSettings settings("LeranEnglish", "InterfaceSouscription");
QString user = settings.value("pseudo").toString();
QString level = settings.value("usuallyLevel").toString();

```

1.6 Interface jeu et Niveau







1.7 Difficulté Rencontrer

Pour le développement du jeu dans qt la prise de mains et la création des interfaces qt sont complexe et demande un travail acharner, de la concentration et la volonté à la rechercher pour le développement des application [QtC++](#).

1.8 Profit Tirer

Le profit tirer pour ce projet est très énorme contenu des langages, les logiciel et technologie utiliser comme. Langage était utiliser pour un but précis comme [PowerAMC](#) pour renforcer mes capacités pour analyse et la modélisation [UML](#), [Qt desing](#) pour l'infographie et le design des application web, [QtC++](#) pour le développement des jeux et application desktop et qui est aussi actuellement beaucoup demander pour la mise en place des systèmes embarqué comme les cartes arduino vue la rapide et le contact direct du langage c++ ou appareil électronique, [TekMaker](#) est un langage des balises pour la rédaction et Édition des articles par exemple.

1.9 Conclusion

Ce projet m'a permit de développer mes compétence en analyse UML, développement d'application C++ et comprendre le principe d'exécution POO avec c++.

avec qtC++, j'ai développé mes connaissance et compétence avec ce framework ide. il dispose d'un système d'exécution unique avec une interface facile pour la prise en mains.

le code source reste disponible sur [github](#) : [blab-user](#) pour les étudiants souhaitant développer un projet similaire avec QtC++, QtPython et le langage de balise Latex.