

Marc Silanus

professeur de Sciences Industrielles de l'Ingénieur
Lycée Alphonse Benoit
84800 L'ISLE SUR LA SORGUE

Mini-Projet Bases de Données

DIU Enseigner l'Informatique au Lycée

Sources : <https://github.com/msilanus/GNL>

Table des matières

1	Objectif du projet.....	3
2	Expression des besoins.....	3
3	Fonctionnalités attendues.....	3
4	Dictionnaire des données.....	4
5	Dépendances fonctionnelles.....	4
6	Modèle conceptuel (MCD).....	5
7	Modèle logique ou schéma (MLD) :.....	5
8	Traduction MySQL.....	5
8.1	Vue des tables.....	5
8.2	Codage des tables.....	6
9	L'application GNL Gestion de Notes Lycée.....	10
9.1	Présentation.....	10
9.2	Architecture de l'application.....	11
9.3	Les classes métier.....	12
9.4	L'IHM.....	14
10	Guide d'utilisation.....	19
10.1	Installation.....	19
10.2	Exécution.....	19
10.3	Utilisation.....	20
11	Conclusion.....	21
12	Remerciements.....	22
13	Webographie.....	22



1 Objectif du projet

Mettre en place une base de données pour la gestion simplifiée des notes d'élèves d'un collège/lycée.

2 Expression des besoins

Plusieurs entretiens ont été réalisés avec des enseignants, des CPE et le proviseur. Il en ressort qu'il faut représenter les concepts de classe (ensemble d'élèves pour une année donnée pour un niveau donné), de professeurs enseignants une ou plusieurs disciplines à différentes classes et d'élèves faisant partie d'une seule classe pour une année donnée et un niveau donné. Chaque discipline donne lieu à une ou plusieurs évaluations et notes à partir desquelles les moyennes par discipline, par trimestre et par année seront calculées.

Aucun coefficient n'est appliqué aux notes.

3 Fonctionnalités attendues

Les fonctionnalités attendues sont les suivantes :

- Ajout de discipline;
- Ajout d'enseignants;
- Ajout d'élèves;
- Créer les classes chaque année pour chaque niveau;
- Affecter les notes d'une discipline donnée par un professeur à un élève;
- Calculer les moyennes et autres statistiques liées aux notes.

4 Dictionnaire des données

Libellé	Désignation	Type	Taille	Remarque
id_classe	Identifiant d'une classe	Alphanumérique	10	Exemple : 2GT1, 1STI2D3, TG4
niveau	Niveau d'une classe	Alphanumérique	10	Seconde, Première, Terminale
annee	Année de la classe	Numérique		
id_professeur	Identifiant d'un professeur	Alphanumérique	13	Numen (ex : 16E0412345ABC)
nom	Nom du professeur	Alphanumérique	30	
prenom	Prénom du professeur	Alphanumérique	30	
id_eleve	Identifiant de l'élève	Alphanumérique	11	INE (ex : 111111111W)
nom	Nom de l'élève	Alphanumérique	30	
prenom	Prénom de l'élève	Alphanumérique	30	
note	Note de l'élève	Numérique		Pour une matière donnée
devoir	Nom du devoir	Alphanumérique	30	Identifie un devoir
date	À une date donnée	Numérique		
id_matiere	Identifiant de la matière	Alphanumérique	30	Identifiant unique (ex : mathN2)
libelle	Libellé de la matière	Alphanumérique	30	Nom de la matière (ex : math)
description	Description textuelle	Texte		Optionnelle

5 Dépendances fonctionnelles

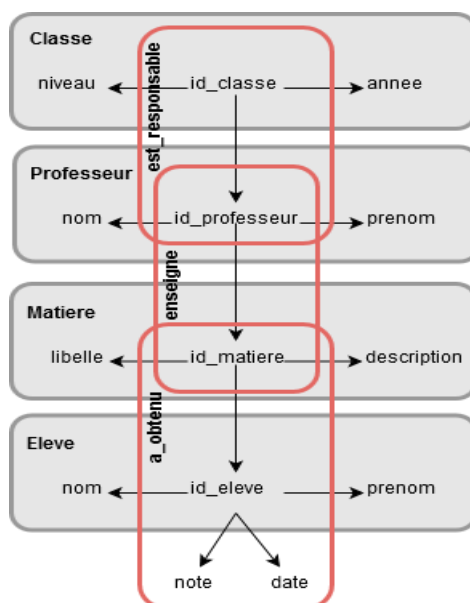


figure 1 - Dépendances fonctionnelles

6 Modèle conceptuel (MCD)

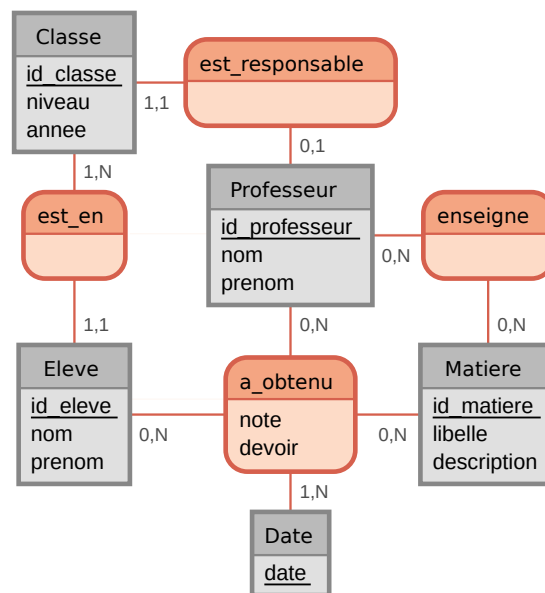


figure 2 - MCD

7 Modèle logique ou schéma (MLD) :

- **Souligné** : clé primaire
- **Italique** : clé étrangère

Classe (id_classe, niveau, annee, *id_professeur*)

Professeur (id_professeur, nom, prenom)

enseigne (id_professeur, *id_matiere*)

Eleve (id_eleve, nom, prenom, *id_classe*)

a_obtenu (id_eleve, *id_professeur*, *id_matiere*, *date*, note, devoir)

Matiere (id_matiere, libelle, description)

8 Traduction MySQL

8.1 Vue des tables

Vue depuis le concepteur graphique de *phpmyadmin* :

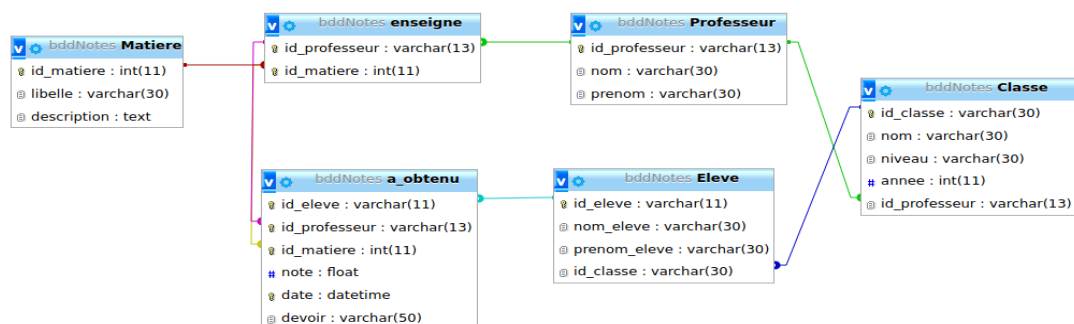


figure 3 - MLD

8.2 Codage des tables

```
--
-- Base de données : `bddNotes`
--
-- -----
--
-- Structure de la table `a_obtenu`
--
CREATE TABLE `a_obtenu` (
  `id_eleve` varchar(11) NOT NULL,
  `id_professeur` varchar(13) NOT NULL,
  `id_matiere` int(11) NOT NULL,
  `note` float DEFAULT NULL,
  `date` datetime NOT NULL,
  `devoir` varchar(50) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- -----
--
-- Structure de la table `Classe`
--
CREATE TABLE `Classe` (
  `id_classe` varchar(30) NOT NULL,
  `nom` varchar(30) NOT NULL,
  `niveau` varchar(30) DEFAULT NULL,
  `annee` int(11) DEFAULT NULL,
  `id_professeur` varchar(13) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- -----
--
-- Structure de la table `Eleve`
--
CREATE TABLE `Eleve` (
  `id_eleve` varchar(11) NOT NULL,
  `nom_eleve` varchar(30) DEFAULT NULL,
  `prenom_eleve` varchar(30) DEFAULT NULL,
  `id_classe` varchar(30) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- -----
```

```
--  
-- Structure de la table `enseigne`  
--  
  
CREATE TABLE `enseigne` (  
  `id_professeur` varchar(13) NOT NULL,  
  `id_matiere` int(11) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
  
-----  
  
--  
-- Structure de la table `Matiere`  
--  
  
CREATE TABLE `Matiere` (  
  `id_matiere` int(11) NOT NULL,  
  `libelle` varchar(30) NOT NULL,  
  `description` text  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
  
-----  
  
--  
-- Structure de la table `Professeur`  
--  
  
CREATE TABLE `Professeur` (  
  `id_professeur` varchar(13) NOT NULL,  
  `nom` varchar(30) DEFAULT NULL,  
  `prenom` varchar(30) DEFAULT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
--  
-- Index pour la table `a_obtenu`  
--  
ALTER TABLE `a_obtenu`  
  ADD PRIMARY KEY (`id_eleve`,`id_professeur`,`id_matiere`,`date`),  
  ADD KEY `id_matiere` (`id_matiere`),  
  ADD KEY `id_professeur` (`id_professeur`);  
  
--  
-- Index pour la table `Classe`  
--  
ALTER TABLE `Classe`  
  ADD PRIMARY KEY (`id_classe`),  
  ADD KEY `id_professeur` (`id_professeur`);  
  
--  
-- Index pour la table `Eleve`  
--  
ALTER TABLE `Eleve`  
  ADD PRIMARY KEY (`id_eleve`),  
  ADD KEY `id_classe` (`id_classe`);  
  
--  
-- Index pour la table `enseigne`  
--  
ALTER TABLE `enseigne`  
  ADD PRIMARY KEY (`id_professeur`,`id_matiere`),  
  ADD KEY `id_matiere` (`id_matiere`);  
  
--  
-- Index pour la table `Matiere`  
--  
ALTER TABLE `Matiere`  
  ADD PRIMARY KEY (`id_matiere`);  
  
--  
-- Index pour la table `Professeur`  
--  
ALTER TABLE `Professeur`  
  ADD PRIMARY KEY (`id_professeur`);
```

```
--  
-- AUTO_INCREMENT pour la table `Matiere`  
--  
ALTER TABLE `Matiere`  
  MODIFY `id_matiere` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=19;
```



```
--  
-- Contraintes pour les tables exportées  
--  
  
--  
-- Contraintes pour la table `a_obtenu`  
--  
ALTER TABLE `a_obtenu`  
  ADD CONSTRAINT `a_obtenu_ibfk_1` FOREIGN KEY (`id_matiere`) REFERENCES  
`enseigne` (`id_matiere`),  
  ADD CONSTRAINT `a_obtenu_ibfk_2` FOREIGN KEY (`id_professeur`) REFERENCES  
`enseigne` (`id_professeur`),  
  ADD CONSTRAINT `a_obtenu_ibfk_3` FOREIGN KEY (`id_eleve`) REFERENCES  
`Eleve` (`id_eleve`);  
  
--  
-- Contraintes pour la table `Classe`  
--  
ALTER TABLE `Classe`  
  ADD CONSTRAINT `Classe_ibfk_1` FOREIGN KEY (`id_professeur`) REFERENCES  
`Professeur` (`id_professeur`);  
  
--  
-- Contraintes pour la table `Eleve`  
--  
ALTER TABLE `Eleve`  
  ADD CONSTRAINT `Eleve_ibfk_1` FOREIGN KEY (`id_classe`) REFERENCES  
`Classe` (`id_classe`);  
  
--  
-- Contraintes pour la table `enseigne`  
--  
ALTER TABLE `enseigne`  
  ADD CONSTRAINT `enseigne_ibfk_1` FOREIGN KEY (`id_matiere`) REFERENCES  
`Matiere` (`id_matiere`),  
  ADD CONSTRAINT `enseigne_ibfk_2` FOREIGN KEY (`id_professeur`) REFERENCES  
`Professeur` (`id_professeur`);
```

9 L'application GNL Gestion de Notes Lycée

9.1 Présentation

L'application **GNL** pour **Gestion de Notes Lycée** est écrite en Python et s'appuie sur la librairie *PyQt5*. L'IHM est dessinée avec *QtDesigner* et se présente comme un module Python importé dans le programme principal.

L'accès aux données, stockées dans une base de données *MySQL*, est assuré par le driver « *MySQL connector* ». Un ensemble de classes métiers spécialise les accès à la base de données.

Le logiciel assure les fonctionnalités suivantes :

- Vue **Administration**

Cette vue assure les opérations de bases comme la création, la modification, l'insertion et la suppression d'une classe, d'une matière, d'un professeur et d'un élève.

- **Gestion des classes**

Filtrage par année et par niveau. L'ajout ou la modification nécessite un nom et un niveau. Une nouvelle classe est associée à l'année en cours.

- **Gestion des matières**

Nécessite une nom de matière. Une description plus précise est optionnelle.

- **Gestion des professeurs**

Nécessite un identifiant constitué du NUMEN du professeur et au minimum son nom.

- **Gestion des élèves**

Nécessite un identifiant constitué de l'INE de l'élève, sa classe d'affectation, son nom et son prénom.

- Vue **Professeur**

Cette vue permet à un professeur de s'identifier (pas de gestion de compte dans cette application) et de gérer ses notes par classe et par devoir.

- Identification du professeur dans la liste des professeurs de la base.
 - Choix de la classe à gérer
 - Choix de la matière à gérer enseignée par le professeur
 - Ajouter ou supprimer un devoir pour la classe et la matière sélectionnées
 - Ajouter ou modifier les notes pour les élèves de la classe pour la matière et le devoir sélectionnés.

- Vue **Association**

Cette vue permet d'associer un professeur aux matières qu'il enseigne

- Identification du professeurs
- Sélection des matières qu'il enseigne

- Vue **Elève**

Cette vue assure la consultation des notes d'un élève.

- Identification d'un élève dans une classe.
- Consultation des moyennes par matières avec moyenne de la classe, note la plus haute et la plus basse.
- Filtrage des notes par matières
- Consultation des notes par devoirs avec moyenne de la classe au devoir, note la plus haute et la plus basse.

9.2 Architecture de l'application

L'application est construite autour de la Classe **ApplicationIHM** qui hérite de la classe **QDialog** générée via **QtDesigner** et qui décrit l'IHM et gère les événements.

Le module **libGestionNotes** contient les classes métier dont les objets sont utilisés dans la classe **ApplicationIHM**.

Ci-dessous, le modèle UML de l'application :

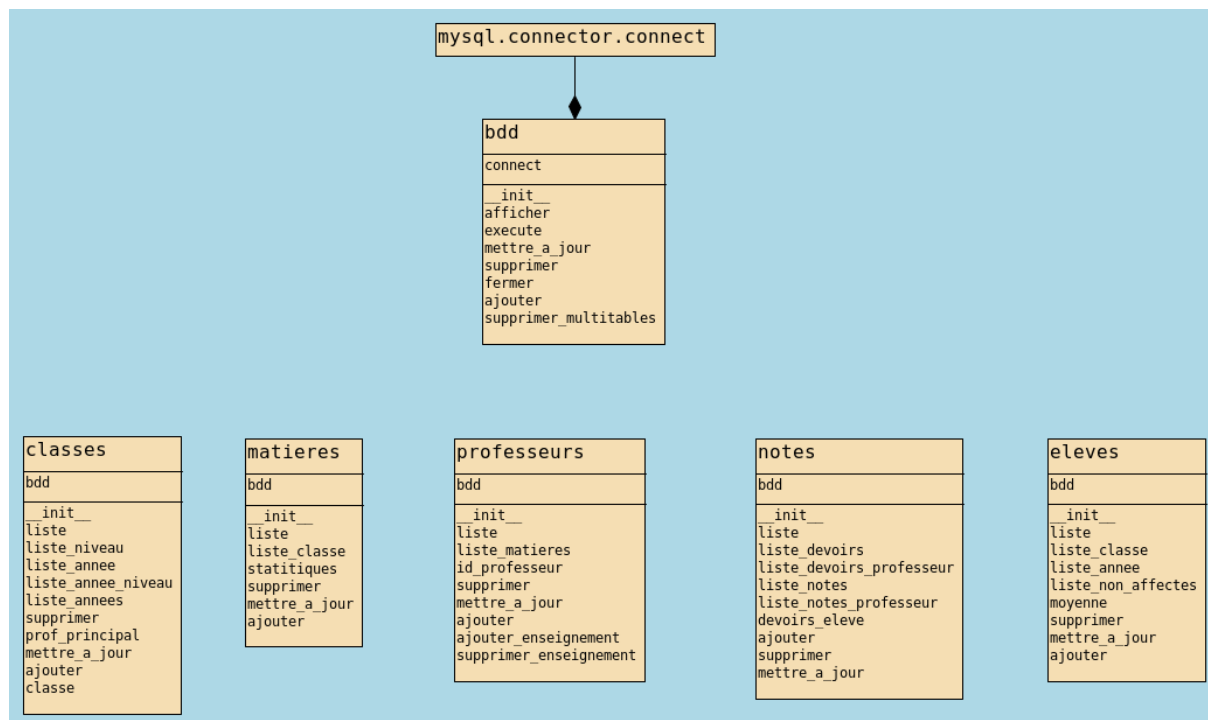


figure 4 – Modèle du module **libGestionNotes**

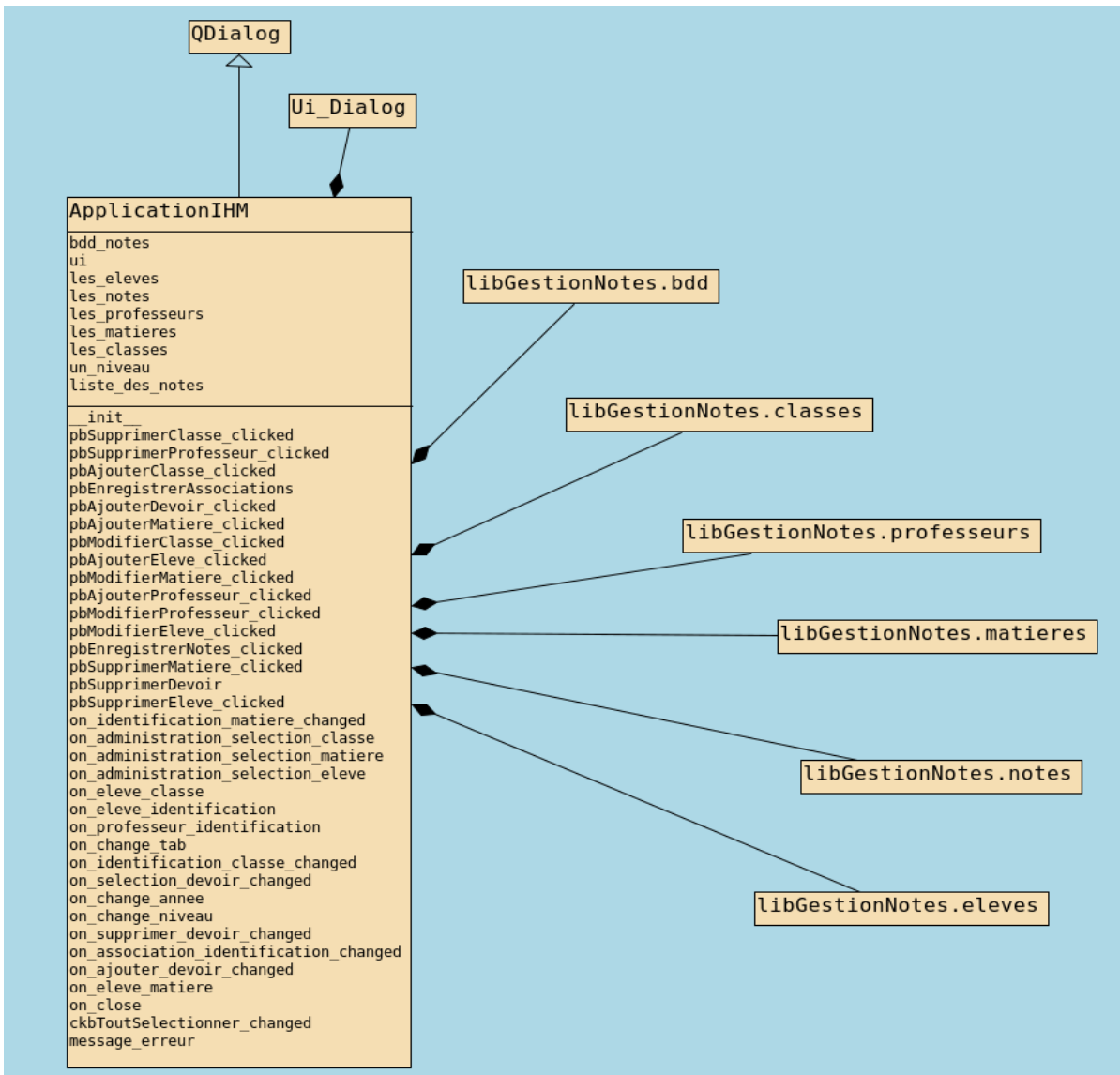


figure 5 - Modèle du module *gestionNotesNSI*

9.3 Les classes métier

Le module **libGestionNotes** contient les classes métier dont les objets sont utilisés dans la classe **ApplicationIHM**.

Ces classes ont pour but de faciliter les opérations sur les tables de la base de données sans se soucier de la construction des requêtes MySQL.

Leurs méthodes principales sont :

- **ajouter()**
- **Supprimer()**
- **mettre_a_jour()**

Ces méthodes sont des spécialisations de celles de la classe **bdd** à qui est déléguée la construction des requêtes et leur soumission au serveur de base de données MySQL.

Cette classe définit les méthodes :

- **`_init(config)`** : Constructeur de la classe, nécessite des paramètres de connexion pour assurer la connexion au serveur MySQL en s'appuyant sur les fonctions du module **`mysql.connector`**.
- **`execute(query)`** : soumission d'une requête au serveur MySQL
- **`afficher(table, colonnes, filtre)`** : Construit, exécute et retourne sous forme d'une liste de dictionnaires le résultat d'une requête de sélection des **colonnes** des **tables** en fonction d'un **filtre**.

Exemple :

```
SELECT Eleve.nom_eleve as nom, Eleve.prenom_eleve as prenom, a_obtenu.devoir as devoir,
a_obtenu.note as note, Matiere.libelle as libelle, Classe.id_classe as classe FROM
`Classe`, `Eleve`, `a_obtenu`, `Matiere` WHERE Eleve.id_eleve=a_obtenu.id_eleve AND
a_obtenu.id_matiere=Matiere.id_matiere AND Classe.id_classe=Eleve.id_classe AND
a_obtenu.id_eleve='38610669270' AND Matiere.id_matiere='8';
```

```
Devoirs : [{ 'nom': 'BEURGUEUR', 'prenom': 'Cochise', 'devoir': 'synthèse de texte',
'note': 4.0, 'libelle': 'Français', 'classe': '1G1_20'}, { 'nom': 'BEURGUEUR', 'prenom':
'Cochise', 'devoir': 'Le Misanthrope', 'note': 13.0, 'libelle': 'Français', 'classe':
'1G1_20'}]
```

- **`supprimer(table, conditions)`** : Construit, exécute et retourne le nombre de lignes affectées par une requête de suppression des données dans **table** suivant certaines **conditions**.

```
query = "DELETE FROM " + table + " WHERE "
for key, value in conditions.items():
    #print(key, value)
    if "_op" in key:
        query += " " + value + " "
    else:
        query += key + "=" + value
query += ";"
print(query)
exec_query = self.execute(query)
return exec_query
```

- **`supprimer_multitables(table, conditions)`** : Supprimer des données présentes dans une table avec **conditions** sur plusieurs **tables**.

```
query = "DELETE "
query += tables[0]
query += " FROM "
nb_tables = len(tables)
for table in tables:
    query += "`" + table + "`"
    if nb_tables > 1:
        query += ", "
        nb_tables -= 1
query += " WHERE "
for key, value in conditions.items():
    if "_op" in key:
        query += " " + value + " "
    else:
        query += key + "=" + value
print(query)
exec_query = self.execute(query)
return exec_query
```

- **mettre_a_jour(tables, a_modifier, conditions)** : Modifier des données déjà présentes dans une table en fonction des **conditions**.

```

query = "UPDATE "+table+" SET "
nb_key = len(a_modifier)
for key, value in a_modifier.items():
    if value == 'NULL':
        query += "`" + key + "`=" + value
    else :
        query += "`"+key+"`='"+value+"'"
    if nb_key>1:
        query += ", "
        nb_key-=1
    else:
        query += " WHERE "
for key, value in conditions.items():
    if "_op" in key:
        query += " "+ value + " "
    else:
        query += "`"+key+"`='"+value+"'"
query += ";"
print(query)
exec_query = self.execute(query)
return exec_query

```

Exemple de modification d'un élève :

Elève à modifier : 1094872327N

```

Nom : ABELOUBA
Prénom : Habib
Classe : 1G2
UPDATE Eleve SET `nom_eleve`='ABELOUBA', `prenom_eleve`='Habib', `id_classe`='1G2_20'
WHERE `id_eleve`='1094872327N';

SELECT Eleve.id_eleve, Eleve.nom_eleve, Eleve.prenom_eleve, Eleve.id_classe, Classe.nom
FROM `Eleve`, `Classe` WHERE Classe.id_classe=Eleve.id_classe AND Eleve.id_classe='1G2_20'
ORDER BY Eleve.nom_eleve ASC ;

```

9.4 L'IHM

L'IHM est dessinée avec *QtDesigner* et se présente comme un module Python importé dans le programme principal.

QtDesigner est l'outil *Qt* pour concevoir et construire des interfaces utilisateur graphiques (GUI) avec *QtWidgets*. Il permet de composer et personnaliser des fenêtres ou boîtes de dialogue selon le principe du « **WYSIWYG** ».

Les widgets et les formulaires créés avec *QtDesigner* s'intègrent de manière transparente avec le code programmé, en utilisant le mécanisme de **signaux** et de **slots** de *Qt*, afin de pouvoir facilement attribuer un comportement aux éléments graphiques.

Toutes les propriétés définies dans *QtDesigner* peuvent être modifiées dynamiquement dans le code.

<https://doc.qt.io/qt-5/qtdesigner-manual.html>

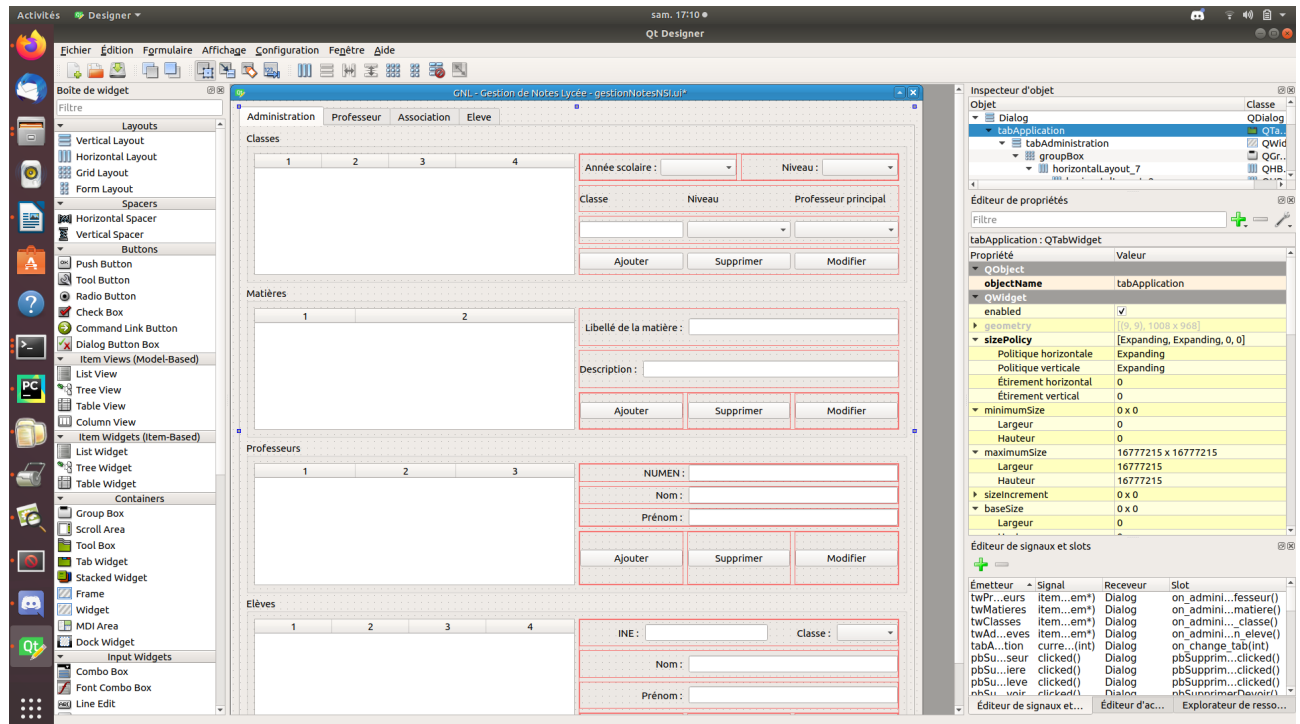


figure 6 - Conception de l'IHM avec **QtDesigner**

L'IHM contient un widget **QtabWidget** dans lequel sont définis les quatre onglets (tab) qui correspondent aux vues **Administration**, **professeur**, **Association** et **Elève**.

Tous les autres widgets sont inclus dans chacun de ces onglets.

Pour présenter les données en tableaux, un widget **QtableWidget** est utilisé. Chacun de ses items est un objet de la classe **QtableWidgetItem** qui dispose de ses propres propriétés.

Des méthodes permettent leur manipulations (afficher, effacer, supprimer, éditer, accéder, ...) et des signaux sont émis lors de certains événements comme leur sélection.

Exemple : Sélection d'un élève dans le **QtableWidget** **twAdministrationEleves**

- Connexion signal/slot :

```
twAdministrationEleves.itemClicked['QTableWidgetItem*'].connect(
    Dialog.on_administration_selection_eleve)
```

- Code du slot **administration_selection_eleve()**:

```
def on_administration_selection_eleve(self, item):
    # Selection de toute la ligne d'un item cliqué
    self.ui.twAdministrationEleves.setRangeSelected( QTableWidgetItemSelectionRange(item.row(),
                                                                                          0,
                                                                                          item.row(),
                                                                                          3),
                                                                                          True)
    ...
    return 0
```

Le code python de l'IHM obtenu après transformation du code XML généré par *QtDesigner* en utilisant l'utilitaire **pyuic5** fourni avec le framework **PyQt5** :

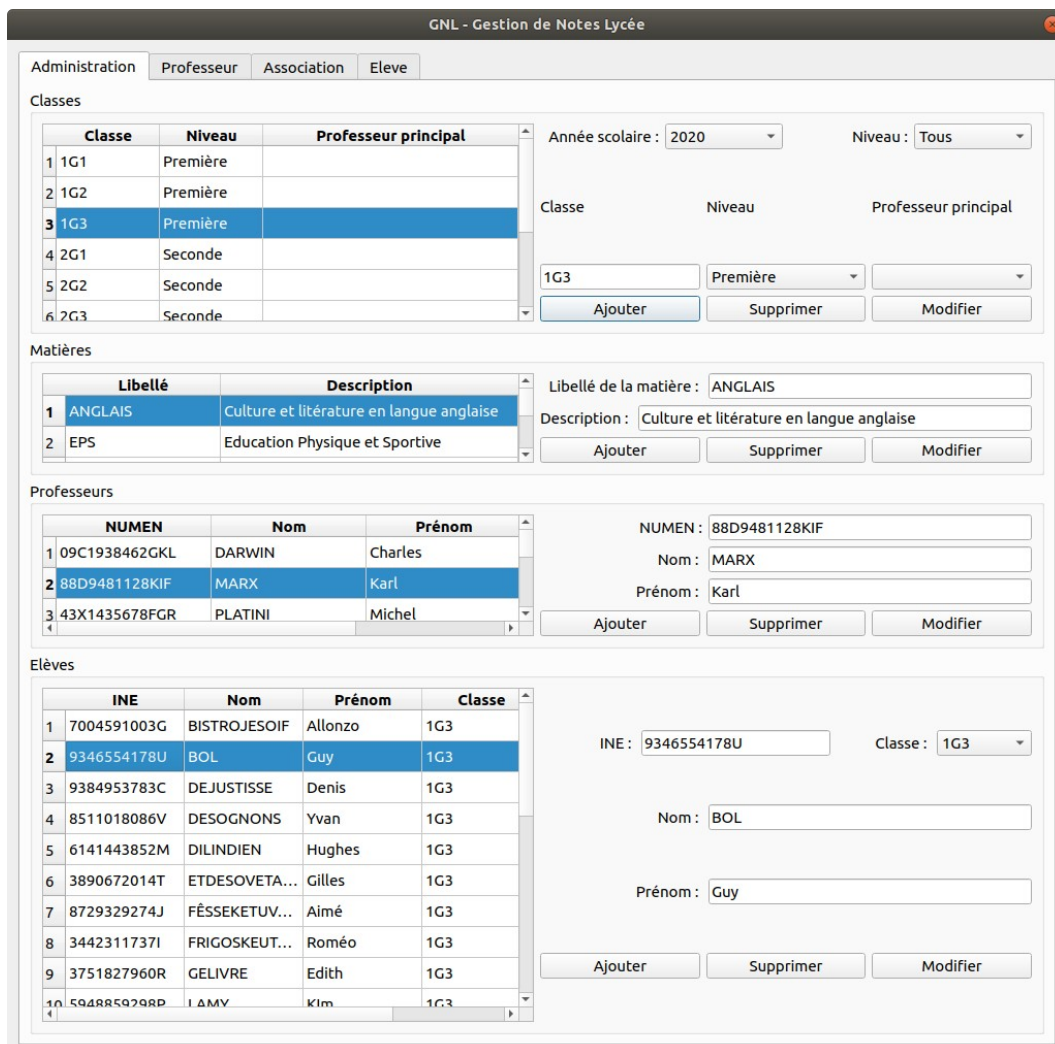
- Entrée : **gestionNotesNSI.ui** : fichier XML généré par QtDesigner
- Sortie : **gestionNotesNSIGUI.py** : module python qui contient le code de l'IHM

```
$ pyuic5 gestionNotesNSI.ui -o gestionNotesNSIGUI.py
```

L'application hérite de la classe **QDialog** et est composée d'un objet de la classe **UI_Dialog** définie dans **gestionNotesNSIGUI.py**. Le code des slots (méthodes) qui correspondent aux réponses à donner aux signaux (événements) en provenance de l'IHM sont définis dans l'application.

Les slots qui commencent par **on_** correspondent à des réponses à une sélection.

Les slots qui commencent par **pb** correspondent à des réponses à un clic sur un bouton.



The screenshot shows the 'Administration' tab of the 'GNL - Gestion de Notes Lycée' application. It contains four main sections: 'Classes', 'Matières', 'Professeurs', and 'Elèves', each with a table and associated form fields.

Classes Section:

Classe	Niveau	Professeur principal
1 1G1	Première	
2 1G2	Première	
3 1G3	Première	
4 2G1	Seconde	
5 2G2	Seconde	
6 2G3	Seconde	

Form fields: Année scolaire: 2020, Niveau: Tous. Buttons: Ajouter, Supprimer, Modifier.

Matières Section:

Libellé	Description
1 ANGLAIS	Culture et littérature en langue anglaise
2 EPS	Education Physique et Sportive

Form fields: Libellé de la matière: ANGLAIS, Description: Culture et littérature en langue anglaise. Buttons: Ajouter, Supprimer, Modifier.

Professeurs Section:

NUMEN	Nom	Prénom
1 09C1938462GKL	DARWIN	Charles
2 88D9481128KIF	MARX	Karl
3 43X1435678FGR	PLATINI	Michel

Form fields: NUMEN: 88D9481128KIF, Nom: MARX, Prénom: Karl. Buttons: Ajouter, Supprimer, Modifier.

Elèves Section:

INE	Nom	Prénom	Classe
1 7004591003G	BISTROJESOIF	Allonzo	1G3
2 9346554178U	BOL	Guy	1G3
3 9384953783C	DEJUSTISSE	Denis	1G3
4 8511018086V	DESOGNONS	Yvan	1G3
5 6141443852M	DILINDIEN	Hughes	1G3
6 3890672014T	ETDESOVETA...	Gilles	1G3
7 8729329274J	FËSSEKETUV...	Aimé	1G3
8 3442311737I	FRIGOSKEUT...	Roméo	1G3
9 3751827960R	GELIVRE	Edith	1G3
10 5948859298P	LAMY	Kim	1G3

Form fields: INE: 9346554178U, Classe: 1G3, Nom: BOL, Prénom: Guy. Buttons: Ajouter, Supprimer, Modifier.

figure 7 – IHM vue Administration

GNL - Gestion de Notes Lycée

Administration Professeur Association Eleve

Identification :

Professeur : Michel PLATINI Classe : 1G1 Matières : EPS

Ajouter/Supprimer un devoir

Devoir à ajouter : Ajouter

Devoir à supprimer : Supprimer

Notes

Sélectionner un devoir : Enregistrer

	Elève	Classe	Matière	Devoir	Date	Note
1	BEURGUEUR Cochise	1G1_20	EPS	Danse classique	2020-02-27 15:11:32	5.0
2	BEURGUEUR Cochise	1G1_20	EPS	foot	2020-02-27 15:12:11	14.0
3	BEURGUEUR Cochise	1G1_20	EPS	javelot	2020-02-27 15:12:30	16.0
4	CAMAN Mehdi	1G1_20	EPS	Danse classique	2020-02-27 15:11:32	12.0
5	CAMAN Mehdi	1G1_20	EPS	foot	2020-02-27 15:12:11	18.0
6	CAMAN Mehdi	1G1_20	EPS	javelot	2020-02-27 15:12:30	10.0
7	CAPET Andy	1G1_20	EPS	Danse classique	2020-02-27 15:11:32	14.0
8	CAPET Andy	1G1_20	EPS	foot	2020-02-27 15:12:11	18.0
9	CAPET Andy	1G1_20	EPS	javelot	2020-02-27 15:12:30	17.0
10	CHANDEL Dino	1G1_20	EPS	Danse classique	2020-02-27 15:11:32	4.0
11	CHANDEL Dino	1G1_20	EPS	foot	2020-02-27 15:12:11	19.0
12	CHANDEL Dino	1G1_20	EPS	javelot	2020-02-27 15:12:30	17.0
13	COT Harry	1G1_20	EPS	Danse classique	2020-02-27 15:11:32	8.0
14	COT Harry	1G1_20	EPS	foot	2020-02-27 15:12:11	16.0
15	COT Harry	1G1_20	EPS	javelot	2020-02-27 15:12:30	3.0
16	DANLGARAJ Mélodie	1G1_20	EPS	Danse classique	2020-02-27 15:11:32	17.0
17	DANLGARAJ Mélodie	1G1_20	EPS	foot	2020-02-27 15:12:11	18.0
18	DANLGARAJ Mélodie	1G1_20	EPS	javelot	2020-02-27 15:12:30	2.0
19	DANSTONCOSTUM...	1G1_20	EPS	Danse classique	2020-02-27 15:11:32	13.0
20	DANSTONCOSTUM...	1G1_20	EPS	foot	2020-02-27 15:12:11	5.0
21	DANSTONCOSTUM...	1G1_20	EPS	javelot	2020-02-27 15:12:30	8.0

figure 8 - IHM vue Professeur

GNL - Gestion de Notes Lycée

Administration Professeur Association Eleve

Identification :

Professeur : Marc SILANUS NUMEN : 16E0412345ABC

Enseigne

☐ Tout Sélectionner/Désélectionner

	Description
<input type="checkbox"/> ANGLAIS	Culture et littérature en langue anglaise
<input type="checkbox"/> EPS	Education Physique et Sportive
<input type="checkbox"/> Français	Lettres modernes
<input type="checkbox"/> HG	Histoire et Géographie
<input type="checkbox"/> MATH	
<input checked="" type="checkbox"/> NSI	Spécialité Numérique et Sciences de l'Informatique en première et terminale
<input type="checkbox"/> Philosophie	Spécialité
<input type="checkbox"/> Sciences Physiques	Physique - Chimie
<input type="checkbox"/> SES	Sciences Economique et Sociale
<input checked="" type="checkbox"/> SI	Spécialité Sciences de l'Ingénieur en première et terminale
<input type="checkbox"/> SVT	Sciences et vie de la Terre

figure 9 - IHM vue Association

GNL - Gestion de Notes Lycée

Administration Professeur Association **Elève**

Identification :

Classe : TG1 Elève : BEALARENVERSE Tom Matières :

Notes

	Matière	Plus basse	Moyenne classe	Plus haute	Moyenne élève
1	ANGLAIS	0.0	11.02	20.0	12.67
2	EPS	0.0	10.67	20.0	6.67
3	Français	0.0	8.95	19.0	11.0
4	MATH	0.0	10.72	20.0	11.8
5	NSI	5.0	12.71	20.0	15.5
6	Philosophie	0.0	10.35	20.0	10.5
7	SES	0.0	10.65	20.0	13.0
8	SI	0.0	9.84	20.0	13.0

figure 10 - IHM vue Elève dans la classe

GNL - Gestion de Notes Lycée

Administration Professeur Association **Elève**

Identification :

Classe : TG1 Elève : BEALARENVERSE Tom Matières : MATH

Notes

	Devoir	Note la plus basse	Moyenne classe	Note la plus haute	Note de l'élève
1	Dérivés	2.0	9.8	20.0	19.0
2	Intégrales	0.0	10.55	20.0	16.0
3	Equa Diff 1er degré	0.0	12.15	20.0	0.0
4	Proba	0.0	10.35	20.0	14.0
5	Statistiques	2.0	10.75	20.0	10.0

figure 11 - IHM vue Elève pour une matière

10 Guide d'utilisation

10.1 Installation

- Cloner le dépôt du projet :

```
$ git clone https://github.com/msilanus/GNL.git
```

- Installer la base de données
 - A partir de **phpmyadmin** : Import → Parcourir → BddNotes.sql
 - A partir du cli MySQL

```
$ sudo mysql
MariaDB [(none)]> source /path/to/GNL/BddNotes.sql
MariaDB [BddNotes3]> show tables;
+-----+
| Tables_in_BddNotes3 |
+-----+
| Classe               |
| Eleve                |
| Matiere              |
| Professeur           |
| a_obtenu              |
| enseigne              |
+-----+
6 rows in set (0.00 sec)
MariaDB [BddNotes3]> GRANT ALL PRIVILEGES ON BddNotes3.* TO <user> WITH
GRANT OPTION;
MariaDB [BddNotes3]> exit ;
```

- Installer les dépendances suivantes :
 - **PyQt5**
 - **mysql-connector**

```
$ pip install PyQt5
$ pip install mysql-connector
```

10.2 Exécution

- Modifier les droits du script pour autoriser le lancement :

```
$ cd /path/to/GNL
$ chmod +x gnl
$ python ./gnl
```

10.3 Utilisation

Le logiciel est fourni avec un jeu de tables contenant des données de démonstration.

- Toutes les classes contiennent des élèves sauf la classe **2G3**
- Tous les professeurs sont associés à une ou plusieurs matières
- Tous les professeurs ont créé des devoirs et mis des notes
- Seul les élèves des classes de 1G1 et TG1 ont des notes

L'utilisation du logiciel se veut suffisamment ergonomique pour ne pas avoir à détailler son fonctionnement.

Attention cependant, la suppression d'une donnée dans une table respecte les contraintes liées aux clés secondaires.

Dans un soucis de démonstration, j'ai choisi de ne pas prévoir le cas de suppression directe, ce qui peut s'avérer un peu lourd à la longue.

Par exemple, il n'est pas possible de supprimer un élève tant qu'il a des notes. Il n'est pas possible de supprimer une classe tant qu'elle contient des élèves, ...

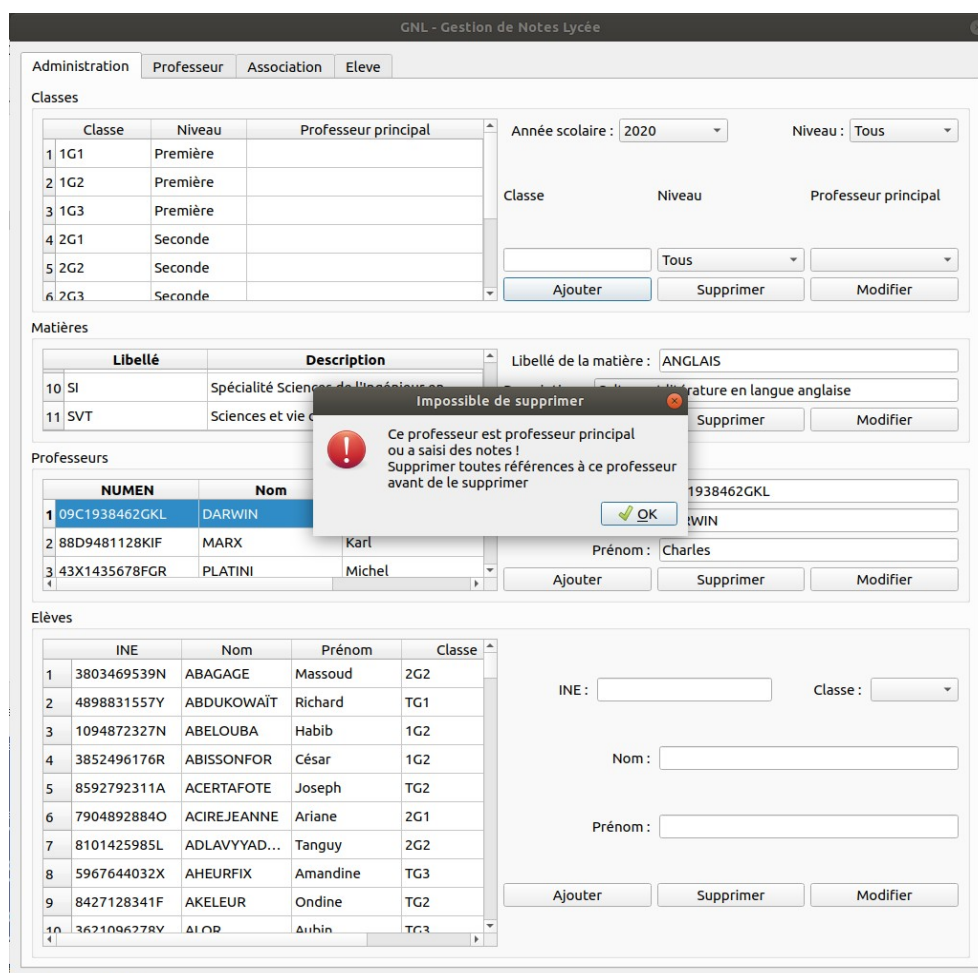


figure 12 - Effet des dépendances fonctionnelles

11 Conclusion

Ce projet m'a permis de me familiariser avec le processus de création de bases de données, l'analyse conduisant à définir les données qui doivent être stockées, définir le format adapté à leur représentation, identifier leur dépendances fonctionnelles et en déduire les tables et les clés primaires et secondaires à utiliser.

L'écriture de requêtes complexes s'est vite avérée fastidieuse et m'a conduit à réfléchir à la réalisation d'un générateur de requêtes. Cela m'a également permis d'approfondir mes modestes connaissances en langage de requête SQL (MySQL), notamment pour les conditions de croisements des tables lors de l'écriture de requêtes complexes.

Exemple : Dans la vue Elève

```
on_eleve_matiere() : Choix de la matière dans la vue élève #####
- Matière sélectionné : Français
- id_matiere : 8

SELECT Eleve.nom_eleve as nom, Eleve.prenom_eleve as prenom, a_obtenu.devoir as devoir,
a_obtenu.note as note, Matiere.libelle as libelle, Classe.id_classe as classe FROM `Classe`,
`Eleve`, `a_obtenu`, `Matiere` WHERE Eleve.id_eleve=a_obtenu.id_eleve AND
a_obtenu.id_matiere=Matiere.id_matiere AND Classe.id_classe=Eleve.id_classe AND
a_obtenu.id_eleve='9832593785Z' AND Matiere.id_matiere='8';

Devoirs : [{ 'nom': 'CAMAN', 'prenom': 'Mehdi', 'devoir': 'synthèse de texte', 'note': 4.0,
'libelle': 'Français', 'classe': '1G1_20' }, { 'nom': 'CAMAN', 'prenom': 'Mehdi', 'devoir': 'Le
Misanthrope', 'note': 19.0, 'libelle': 'Français', 'classe': '1G1_20' }]

synthèse de texte

SELECT COUNT(note) as nombre_notes, AVG(note) as moyenne, MAX(note) as max, MIN(note) as min FROM
`a_obtenu`, `Eleve` WHERE a_obtenu.`id_eleve`=Eleve.id_eleve AND Eleve.id_classe='1G1_20' AND
id_matiere='8' AND devoir='synthèse de texte';

Stat : { 'nombre_notes': 20, 'moyenne': 9.6, 'max': 20.0, 'min': 0.0 }

Le Misanthrope

SELECT COUNT(note) as nombre_notes, AVG(note) as moyenne, MAX(note) as max, MIN(note) as min FROM
`a_obtenu`, `Eleve` WHERE a_obtenu.`id_eleve`=Eleve.id_eleve AND Eleve.id_classe='1G1_20' AND
id_matiere='8' AND devoir='Le Misanthrope';

Stat : { 'nombre_notes': 20, 'moyenne': 10.55, 'max': 19.0, 'min': 1.0 }
```

Exemple : Ajout ou mise à jour de la note d'un élève :

```
##### pbEnregistrerNotes_clicked() : Bouton Enregistrer notes cliqué #####

INSERT INTO a_obtenu (`id_eleve`,`id_professeur`,`id_matiere`,`note`,`date`,`devoir`) VALUES
('38610669270','09C1938462GKL','16','10.0','2020-02-27 14:18:23','Dissection');

1062 (23000): Duplicate entry '38610669270-09C1938462GKL-16-2020-02-27 14:18:23' for key
'PRIMARY'

UPDATE a_obtenu SET `note`='10.0' WHERE `id_professeur`='09C1938462GKL' AND
`id_eleve`='38610669270' AND `id_matiere`='16' AND `devoir`='Dissection' AND `date`='2020-02-27
14:18:23';

Ajout : 38610669270 - BEURGUEUR - Cochise - 1G1_20 - id_matiere : 16 : 10.0 - 2020-03-21 23:04:49
```

12 Remerciements

Nous remercions tous les formateurs qui nous ont permis d'approfondir nos connaissances en base de données et leur utilisation au travers d'un programme python, pour leur patience et disponibilité.

13 Webographie

- <http://www.mocodo.net/>
- https://www.w3schools.com/python/python_mysql_getstarted.asp
- <https://www.riverbankcomputing.com/static/Docs/PyQt5/index.html>
- <https://pynsource.com/>