



PROJET INF3036 - BASE DE DONNÉES

**THÈME: "IMPLÉMENTATION D'UNE APPLICATION WEB
DE GESTION DES EMPLOIS DE TEMPS EN
FACULTÉ DES SCIENCES DE L'UYI"**

RAPPORT FINAL DU PROJET

MEMBRES DU GROUPE (Groupe N°31 SDD)

NOMS ET PRENOMS	MATRICULE
DJEUMEZA DONGMO Julie Merveille	19M2105
MADJOU Alvine Patricia	19M2103
NAKAM YOPDUP Manuella Kristeva	19M2233
NOUCHEN TCHAMBA Parnell Voltaire	19M2326

PLAN DÉTAILLÉ

Introduction

I- Présentation de l'application:

1. Description
2. Objectifs
3. Intérêts

II- Analyse conceptuelle:

1. Besoins fonctionnels
2. Besoins non fonctionnels
3. Acteurs et cas d'utilisation

III- Conception détaillée:

1. Mcd et Mld
2. Diagramme des classes
3. Diagramme des cas d'utilisation

IV- Implémentation:

1. Technologies utilisées
2. Requêtes sql et résultats correspondants
3. Interfaces utilisateurs

Conclusion

Introduction

Dans les établissements tout comme dans toute autre activité entropique, une bonne organisation des activités est une clé de la réussite. Dans le cadre éducatif en particulier, la transmission effective des connaissances et le suivi des élèves/étudiants qu'il comporte en fonction du temps disposé représente l'une de ses principales priorités. Ceci étant, une planification rigoureusement conforme et fiable en ce qui concerne l'occupation des salles, les enseignants programmés pour chaque cours et les horaires auxquelles seront dispensés les cours aux différentes classes devient indispensable pour le bon fonctionnement de la structure: D'où la mise sur pied d'un emploi de temps. C'est dans cette optique qu'intervient notre projet, où le cadre éducatif ici concerné est la faculté des sciences de l'université de Yaoundé I. Pour se faire, nous allons repartir notre travail en quatre principales parties que nous développerons succinctement dans nos prochaines lignes.

I- Présentation de l'application:

1. Description:

Notre projet porte sur la gestion des emplois de temps de la faculté des sciences de l'université de Yaoundé I . À l'issu de celui-ci , une application web sera obtenue à partir de laquelle tout étudiant et enseignant de la facsciences auront la possibilité de consulter et/ou télécharger leur emploi de temps, celui-ci sera bien structuré et surtout fiable.

2. Objectifs:

Vue le nombre assez important de classes que regorge la facscience de l'université de Yaoundé I , il devient difficile de générer tout l'ensemble des emplois de temps manuellement et obtenir quelque chose d'optimal ,de bien structuré et fiable. C'est ainsi que l'application qui résultera de notre projet a pour objectif de pallier à ce problème à travers les différentes fonctionnalités qu'elle regorge :

- Générer les emplois de temps selon différentes vues (enseignant, classe, salle,..)
- Permettre au gestionnaire de l'application de créer un emploi de temps particulier d'une classe en fonction des heures libres encore disponibles
- Gérer l'allocation des amphis/salles à des classes en fonction de leur effectif afin que les étudiants puissent faire cours de manière confortable
- Gérer les horaires de cours dans les amphis/salles afin d'éviter des chevauchements.
- Modifier les horaires de cours d'un enseignant ou d'une classe si besoin se pose sans toutefois engendrer des perturbations dans l'emploi de temps général.

3. Intérêts:

Cette application sera intéressante tant pour l'étudiant, pour l'enseignant que pour la gestion et l'organisation des salles de cours:

- **Coté étudiant:** Grace à cette application, tout étudiant convenablement inscrit à la faculté des sciences de l'université de Yaoundé I, aura la possibilité de consulter l'emploi de temps concernant un cours ,sa filière et/ou spécialité. De même, il pourra être en contact avec l'emploi de temps concernant tous les enseignants de sa filière et/ou de sa spécialité et même ceux du département auquel il appartient, et donc pourra aisément le contacter en cas de besoin.
- **Coté enseignant:** Cette plateforme sera profitable pour tout enseignant de la faculté des sciences de l'UYI, en ce sens où elle lui offrira la possibilité d'être informé sur la disponibilité ou non d'une salle avant de programmer un rattrapage de cours ou dispenser un cours à une classe. Ce qui évitera des chevauchements de cours dans une salle a un instant donné.

II- Analyse conceptuelle:

L'analyse conceptuelle consiste à identifier et définir de manière concrète les différentes fonctionnalités primaires et secondaires que l'application est appelée à effectuer.

1. Besoins fonctionnels:

Il s'agit de l'ensemble des besoins ou fonctionnalités de base, nécessaires que l'application sera amenée à exécuter:

- **La personnalisation d'un emploi de temps:** Le programmeur d'emploi de temps, c'est-à _dire un membre à part entière de l'administration de la faculté, programme un emploi de temps pour une classe précise. Pour cela, deux méthodes lui sont proposées:
 - > **La méthode par formulaire:** A partir de laquelle il renseigne les différents champs(nom de classe, nom de/des enseignant/s, heure de début, heure de fin, nom du cours ...) utiles pour générer un emploi de temps à ne classe donnée.
 - > **La méthode par tableau:** A partir d'un tableau préétabli, exemple d'un emploi de temps, comportant les jours de la semaine, les horaires générales et des espaces alloués pour renseigner directement le cours qui sera dispensé ainsi que l'enseignant devant le dispenser.
- **Le téléchargement d'un emploi de temps:** Une fois que l'administrateur a terminé d'établir un emploi de temps, il a la possibilité de le télécharger directement pour une éventuelle diffusion ou autre besoin.

- **L'enregistrement des cours dans la base de données:** L'application offre la possibilité d'enregistrer un cours pour une classe via un formulaire qui est soumis à l'administrateur. Le cours est donc par la suite sauvegardé dans une base de données. Ainsi, toute modification concernant ledit cours sera également sauvegardée; de cette façon, toutes les informations sur le cours sont gardées.
- **L'enregistrement des enseignants dans la base de données:** FS SCHEDULER offre la possibilité d'enregistrer un enseignant. Les informations de celui-ci seront stockées dans une base de données.
- **L'enregistrement des classes dans la base de données:** L'application offre également la possibilité d'enregistrer une classe. La classe étant stockée sous forme de *Spécialité_Niveau*.
- **L'enregistrement des salles dans la base de données:** Aussi, notre application sauvegardera dans une base de données des salles de la faculté des sciences, à l'intérieur desquelles seront dispensés des cours.
- **L'ajout ou la modification d'un enseignant dans un emploi de temps:** L'administrateur a la possibilité d'insérer un enseignant dans un emploi de temps ou alors, si déjà existant, modifier les informations relatifs à celui-ci.
- **La modification d'une horaire de cours:** Pour une raison ou une autre, il peut arriver que l'on veuille modifier une horaire de cours.
- **La modification d'une salle de cours et des jours de cours** seront aussi rendus possible dans notre application.

2. Besoins non fonctionnels:

Outre les besoins fonctionnels qui sont vus comme des besoins primaires d'une application, les **besoins non fonctionnels** quant à eux peuvent être considérés comme besoins secondaires. Il s'agit en effet des fonctionnalités représentant des indicateurs de qualité de l'exécution des **besoins fonctionnels**. FS SCHEDULER, notre application en a quelques uns:

- **L'authentification:** Afin d'effectuer une modification quelconque sur un emploi de temps, il faut s'authentifier.
- **Privatiser l'emploi de temps d'un enseignant:** Dans cette application, tout le monde a la possibilité de visualiser l'emploi de temps d'un enseignant quelconque. Il devient donc impératif de limiter l'accès aux emplois de temps des enseignants.

3. Acteurs et cas d'utilisation:

Le principal acteur de l'application est **l'administrateur**, un membre de l'administration de la faculté des sciences du campus (Recteur, doyen de la faculté, Chef de département,...). C'est lui qui est en relation directe avec le système. Et donc en plus de pouvoir consulter l'application comme simple utilisateur ou un lambda étudiant de la faculté ou même encore un enseignant, lui seul a la possibilité de créer un emploi de temps, en définissant les jours de cours, les cours à dispenser, les enseignants qui devront les dispenser, les salles dans lesquelles seront dispensés les cours, la classe concernée par le cours et la plage horaire durant laquelle sera tenu le cours. De même, il a la possibilité de modifier un emploi de temps de manière à ce qu'il soit conforme et convenable.

Les acteurs secondaires sont les **étudiants** et les **enseignants** de la faculté, qui ont beaucoup plus un rôle de visiteur ou consultant.

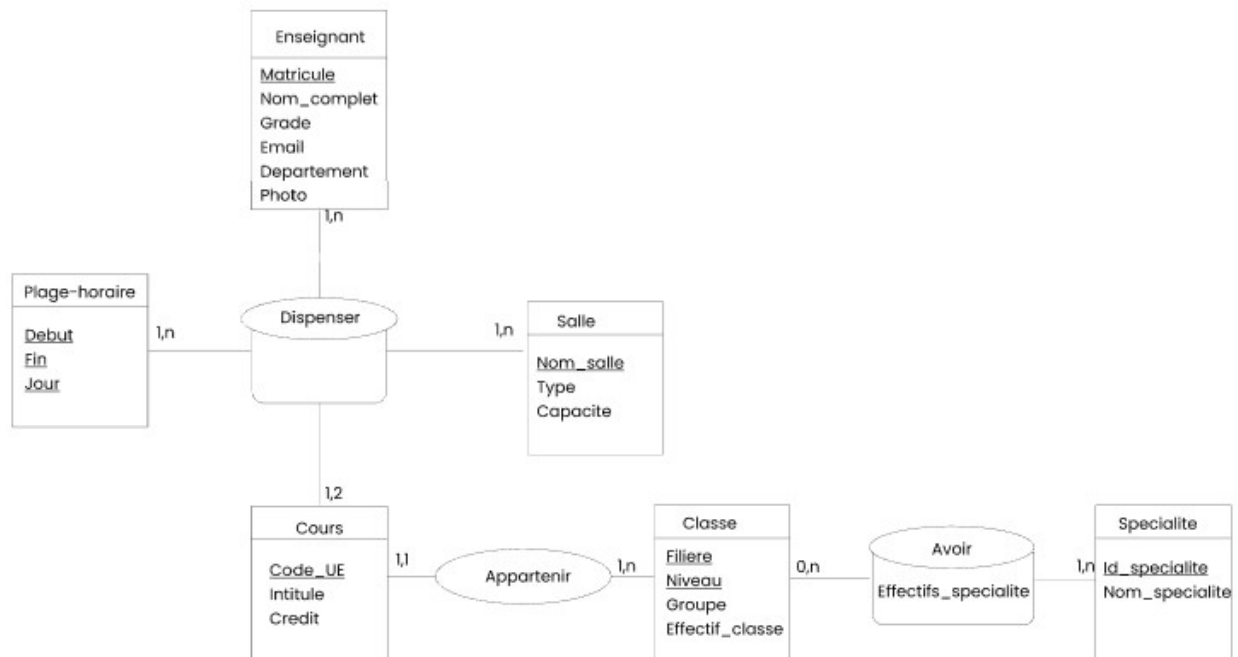
III- Conception détaillée:

Cette partie concerne les niveaux de détails de la conception permettant d'illustrer tous les éléments de conception nécessaires à la réalisation de l'application. Dans notre cas, elle s'appuie sur les modèles conceptuels et logiques des données, les diagrammes de classes et ceux des cas d'utilisation.

1. Mcd et Mld:

- **MCD**

A partir des règles de gestion des données on a pu définir le modèle entité-association décrivant l'aspect conceptuel des données. Ainsi le modèle conceptuel obtenu est le suivant:



MCD de l'application FS SCHEDULER

- **MLD:**

Pour une meilleure analyse du projet, il est nécessaire de construire son modèle logique qui est une représentation logique des données. Ainsi, à partir du modèle conceptuel obtenu précédemment on peut avoir le modèle relationnel normalisé suivant:

Plage_Horaire(*Jour*, *Début*, *Fin*)

Enseignant(*Matricule*, Nom_complet, Grade, Émail, Département, Photo)

Cours(*Code_UE*, Intitule, Crédit)

Classe(*Filière*, *Niveau*, Groupe, Effectif_Classe)

Spécialité(*Id_Spécialité*, Nom_spécialité)

Salle(*Nom_Salle*, Type, Capacité, #*Filière*, #*Niveau*)

Avoir(*Filière*, *Niveau*, *Id_Spécialité*, Effectif_Spécialité)

Dispenser(*Matricule*, *Jour*, *Début*, *Fin*, *Code_UE*, *Filière*, *Niveau*)

A noter que les clés primaires sont données en italique et soulignées tandis que les clés étrangères sont en italique précédées d'un #.

2. Diagramme des classes:

Le diagramme de classes est un schéma qui modélise les différentes classes et les relations entre elles. Le diagramme de classes de notre application est présenté comme suit:

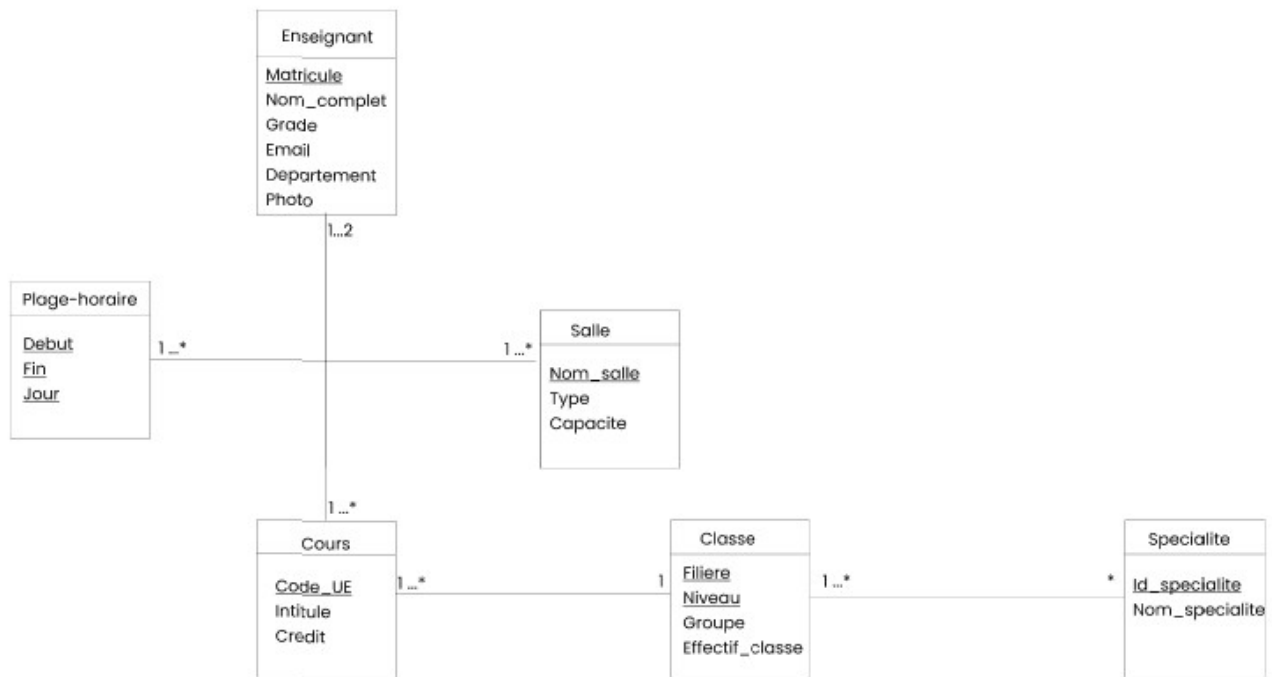
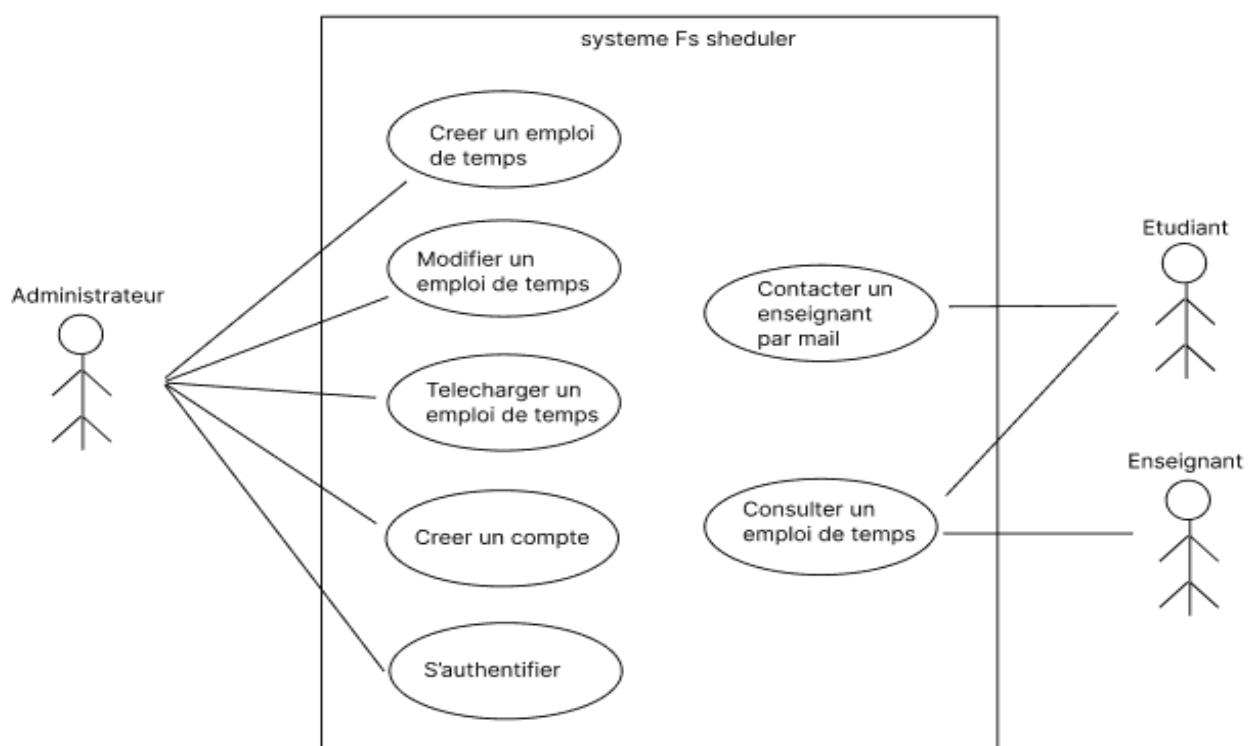


Diagramme de classes de l'application FS SCHEDULER

3. Diagramme des cas d'utilisation:

Le diagramme des cas d'utilisation représente l'interaction entre un acteur et le système. Concernant notre application, l'interaction entre le système FS SCHEDULER et le système est illustré de la manière suivante:



IV- Implémentation:

1. Technologies utilisées:

Les technologies utilisées représentent l'ensemble des outils mis en place pour l'implémentation de l'application.

- **Figma:** Pour le design des différentes pages
- **MySQL-WorkBench** : Comme Système de gestion des bases de données
- **Sql:** Langage d'implémentation de requêtes à la base de données
- **HTML** : pour l'implémentation des pages web
- **CSS** : pour l'embellissement de ces pages web
- **JavaScript** : pour l'interaction entre ces pages
- **Framework Django** : pour l'implémentation du back-end de l'application
- **Quelques bibliothèques python** : pour d'éventuels traitements spécifiques
- **Gitlab:** Outil de versioning pour la sauvegarde des différentes versions du projet

2. Requêtes sql et résultats correspondants:

Une requête désigne une interrogation à la base données. Les différentes requêtes effectuées dans notre application ont été effectuées dans le but de remplir la base de données ou alors d'obtenir une information particulière de celle-ci. On peut avoir les requêtes suivantes :

- Requête d'insertion d'une salle :

```
with connection.cursor() as cursor:
```

```
    #requête SQL
    sql = "INSERT INTO core_salle (nom_salle, type, capacity)\
          VALUES( %s, %s, %s)"

    #les valeurs de la requête SQL
    value = (nom_salle, type, capacity)

    #exécuter le curseur avec la méthode execute() et transmis la requête SQL
    return cursor.execute(sql, value)
```

- Requête d'insertion d'une classe :

```
with connection.cursor() as cursor:
```

```
    # creation de la row specialite
    # si la specialite entree n'existait pas encore, je la cree d'abord
    if not Specialite.objects.filter(nom_specialite = nom_specialite).exists() and nom_specialite != "":

        #requête SQL
        sql = """INSERT INTO core_specialite (nom_specialite)
              VALUES( %s )"""

        #les valeurs de la requête SQL
        value = (nom_specialite,)

        #exécuter le curseur avec la méthode execute() et transmettre la requête SQL
        cursor.execute(sql, value)

    # creation de la row classe

    #requête SQL
    # je recupere d'abord l'id de la specialite que je viens de creer
    print(type(nom_specialite))
    if not nom_specialite == "":
        cursor = connection.cursor()
        query = """select * from core_specialite where nom_specialite = %s"""
        # set variable in query
        cursor.execute(query, (nom_specialite,))
        # fetch result
        id_specialite = cursor.fetchall()[0][0] # pour ne prendre que l'id de la specialite
    else:
        id_specialite = None

    # maintenant je cree la classe en question
    sql = "INSERT INTO core_classe (filier, niveau, groupe, effectif, specialite_id)\
          VALUES( %s, %s, %s, %s, %s)"

    #les valeurs de la requête SQL
    value = (filier, niveau, groupe, effectif, id_specialite)

    #exécuter le curseur avec la méthode execute() et transmettre la requête SQL
    return cursor.execute(sql, value)
```

- Requête d'insertion d'un enseignant :

```
with connection.cursor() as cursor:
```

```
    #requête SQL
    sql = "INSERT INTO core_teacher (matricule, email, full_name, degree, department, pp)\
          VALUES( %s, %s, %s, %s, %s, %s)"

    #les valeurs de la requête SQL
    value = (matricule, email, full_name, degree, department, pp)

    #exécuter le curseur avec la méthode execute() et transmis la requête SQL
    return cursor.execute(sql, value)
```

- Requête d'insertion d'un cours :

```
#requête SQL
sql = "INSERT INTO core_courses (code_ue, libele, credit, classe_id, main_teacher_id, second_teacher_id)\
      VALUES( %s, %s, %s, %s, %s, %s)"

#les valeurs de la requête SQL
value = (code_ue, libele, credit, id_class, id_second_teacher, id_main_teacher)

#exécuter le curseur avec la méthode execute() et transmis la requête SQL
return cursor.execute(sql, value)
```

- Requête d'ajout d'un emploi de temps :

```
with connection.cursor() as cursor:

    cursor = connection.cursor()

    #requête SQL
    sql = "INSERT INTO core_schedulers (is_td, course_id, plage_id, salle_id, teacher1_id, teacher2_id)\
          VALUES( %s, %s, %s, %s, %s, %s)"

    #les valeurs de la requête SQL
    value = (is_td, self.id_course, self.id_plage, self.id_salle, self.id_teacher1, self.id_teacher2)

    #exécuter le curseur avec la méthode execute() et transmis la requête SQL
    return cursor.execute(sql, value)
```

- Requête d'obtention d'un emploi de temps d'une salle :

```
for i in programmations:
```

```
    # on recupere l'enseignant1
    query = """select full_name, degree from core_teacher where id = %s"""
    cursor.execute(query, (i[0],))
    teacher1 = cursor.fetchall()[0]
    if teacher1[1][0] == 'P' or teacher1[1][0] == 'c' or teacher1[1][0] == 'C':
        nom_ens1 = 'Pr '+teacher1[0]
    if teacher1[1][0] == 'A' or teacher1[1][0] == 'S':
        nom_ens1 = 'Dr '+teacher1[0]
    if teacher1[1][0] == 'V':
        nom_ens1 = 'M. '+teacher1[0]

    # on recupere l'enseignant2
    if not i[1] == None:
        query = """select full_name, degree from core_teacher where id = %s"""
        cursor.execute(query, (i[1],))
        teacher2 = cursor.fetchall()[0]
        if teacher2[1][0] == 'P' or teacher2[1][0] == 'c' or teacher2[1][0] == 'C':
            nom_ens2 = 'Pr '+teacher2[0]
        if teacher2[1][0] == 'A' or teacher2[1][0] == 'S':
            nom_ens2 = 'Dr '+teacher2[0]
        if teacher2[1][0] == 'V':
            nom_ens2 = 'M. '+teacher2[0]
    else:
        nom_ens2 = ""

    # on recupere le cours
    query = """select code_ue from core_courses where id = %s"""
    cursor.execute(query, (i[2],))
    course = cursor.fetchall()[0][0]
    course = str(course)

    # on recupere la classe
    query = """select classe_id from core_courses where id = %s"""
    print(i[2], ".....")
    cursor.execute(query, (i[2],))
    classe_id = cursor.fetchall()[0][0]

    query = """select filiere, niveau, specialite_id, groupe from core_classe where id = %s"""
    cursor.execute(query, (classe_id,))
    classe = cursor.fetchall()[0]

    |

    globals()['emploi%s' % i[-1]] = Emploi()
    globals()['emploi%s' % i[-1]].nom_ens1 = nom_ens1
    globals()['emploi%s' % i[-1]].nom_ens2 = nom_ens2
    globals()['emploi%s' % i[-1]].classe = classe
    globals()['emploi%s' % i[-1]].cours = course
```

- Requête d'obtention d'un emploi de temps d'une classe :

```
for i in programmations:
```

```

    # on recupere l'enseignant1
    query = """select full_name, degree from core_teacher where id = %s"""
    cursor.execute(query, (i[0],))
    teacher1 = cursor.fetchall()[0]
    if teacher1[1][0] == 'P' or teacher1[1][0] == 'c' or teacher1[1][0] == 'C':
        nom_ens1 = 'Pr '+teacher1[0]
    if teacher1[1][0] == 'A' or teacher1[1][0] == 'S':
        nom_ens1 = 'Dr '+teacher1[0]
    if teacher1[1][0] == 'V':
        nom_ens1 = 'M. '+teacher1[0]

    # on recupere l'enseignant2
    if not i[1] == None:
        query = """select full name, degree from core_teacher where id = %s"""
        cursor.execute(query, (i[1],))
        teacher2 = cursor.fetchall()[0]
        if teacher2[1][0] == 'P' or teacher2[1][0] == 'c' or teacher2[1][0] == 'C':
            nom_ens2 = 'Pr '+teacher2[0]
        if teacher2[1][0] == 'A' or teacher2[1][0] == 'S':
            nom_ens2 = 'Dr '+teacher2[0]
        if teacher2[1][0] == 'V':
            nom_ens2 = 'M. '+teacher2[0]
    else:
        nom_ens2 = ""

    # on recupere le cours
    query = """select code ue from core_courses where id = %s"""
    cursor.execute(query, (i[2],))
    course = cursor.fetchall()[0][0]
    course = str(course)

    # on recupere la salle
    query = """select nom_salle from core_salle where id = %s"""
    cursor.execute(query, (i[3],))
    salle = cursor.fetchall()[0][0]
    salle = str(salle)

    globals()['emploi%s' % i[-1]] = Emploi()
    globals()['emploi%s' % i[-1]].nom_ens1 = nom_ens1
    globals()['emploi%s' % i[-1]].nom_ens2 = nom_ens2
    globals()['emploi%s' % i[-1]].cours = course
    globals()['emploi%s' % i[-1]].salle = salle

```

- Requête d'obtention d'un emploi de temps d'un enseignant :

```

for i in programmations:

    # on recupere le cours
    query = """select code_ue from core_courses where id = %s"""
    cursor.execute(query, (i[1],))
    course = cursor.fetchall()[0][0]
    course = str(course)

    # on recupere la salle
    query = """select nom_salle from core_salle where id = %s"""
    cursor.execute(query, (i[0],))
    salle = cursor.fetchall()[0][0]
    salle = str(salle)

    # on recupere la classe
    query = """select classe_id from core_courses where id = %s"""
    print(i[1], ".....")
    cursor.execute(query, (i[1],))
    classe_id = cursor.fetchall()[0][0]

    query = """select filiere, niveau, specialite_id, groupe from core_classe where id = %s"""
    cursor.execute(query, (classe_id,))
    classe = cursor.fetchall()[0]

    if not classe[2] == None:
        query = """select nom_specialite from core_specialite where id = %s"""
        cursor.execute(query, (classe[2],))
        specialite = cursor.fetchall()[0][0]
    else:
        specialite = ""

    if not specialite == "":
        if classe[3] == "":
            classe = str(classe[0])+" "+str(classe[1])+" (" +str(specialite)+") "
        else:
            classe = str(classe[0])+" "+str(classe[1])+" (" +str(specialite)+") groupe "+ str(classe[3])
    else:
        if classe[1] == 'L3' or classe[1] == 'M1':
            if classe[3] == "":
                classe = str(classe[0])+" "+str(classe[1])+" (tronc commun) "
            else:
                classe = str(classe[0])+" "+str(classe[1])+" (tronc commun) groupe "+ str(classe[3])
        else:
            if classe[3] == "":
                classe = str(classe[0])+" "+str(classe[1])
            else:
                classe = str(classe[0])+" "+str(classe[1])+" groupe "+ str(classe[3])

    globals()['emploi%s' % i[-1]] = Emploi()
    globals()['emploi%s' % i[-1]].classe = classe
    globals()['emploi%s' % i[-1]].salle = salle
    globals()['emploi%s' % i[-1]].cours = course

```

3. Interfaces utilisateurs:


Les interfaces utilisateurs sont les différentes vues présentant les différentes pages de l'application. Les interfaces de notre application sont données en captures ainsi qui suit :

- **Page d'authentification** : C'est la page de connexion de l'administrateur.

FS SCHEDULER

HomePlanningLog in

Log in Page



doyen

.....

☐Remember me
 [Forgot password?](#)

LOGIN

Our address

Quick access

Newletter

- **Page d'accueil** : Comme son nom l'indique, c'est la page d'accueil de l'application. C'est celle qui s'affiche au premier abord avec l'application :

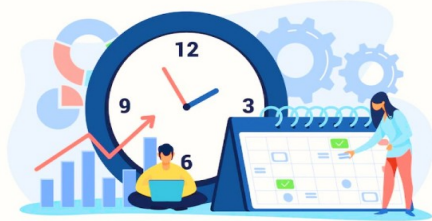
FS SCHEDULER

HomePlanningLog in

My school Time table

Organized courses for better success !!

Planning



Concerning the concerned school

Created by decree N°93/036 of January 29, 1993, the university of Yaounde I is a public ,scientist and cultural establishment with legal personality and financial autonomy. It is places under the supervision of ministry in charge of higher education.

In abbreviated UY1, this established regroupes articularly 03 faculties.


1. Faculty of arts letters and humanities

2. Faculty of sciences

3. Faculty of education sciences

STUDENTS	TEACHERS
Any person registered and recognized as a students of the university of Yaounde I belonging to a specific field of study and/or speciality and regularly attending classes in the said institution	Any person registered and recognized as a students of the university of Yaounde I belonging to a specific field of study and/or speciality and regularly attending classes in the said institution

Who can use it?



THE ADMINISTRATOR

Only the administrator (or certain members of the administration including a dean in the case of vacancy of the rector) is called to modify the schedules. The others (teachers, students, security staff) can only consult and download.

Our address
Location : Yaounde
Tel : 699 65 88 65 / 673 56 45 21
E-mail : fsscheduler@gmail.cm


Quick access
Home
About us
Contact

Newletter
Subscribe to our newsletter to stay informed about application!
E-mail address :

copyright 2022 | FS Scheduler | Web application

- **Page des salles :** C'est la page qui contient la liste des salles de la faculté des sciences.

FS SCHEDULER
Home
Planning
Log in



Classrooms		Classes		Teachers	
A502 (Amphi) 100 places Get the scheduler	A1002 (Classroom) 1200 places Get the scheduler	A1001 (Classroom) 1100 places Get the scheduler	A501 (Classroom) 100 places Get the scheduler	A135 (Classroom) 100 places Get the scheduler	A250 (Classroom) 100 places Get the scheduler
S006 (Classroom) 100 places Get the scheduler	S111 (Classroom) 100 places Get the scheduler	S008 (Classroom) 120 places Get the scheduler	S005 (Classroom) 100 places Get the scheduler	E204 (Classroom) 100 places Get the scheduler	R110 (Classroom) 500 places Get the scheduler
R108 (Classroom) 150 places Get the scheduler					

Our address
Location : Yaounde

Quick access
Home

Newletter
Subscribe to our newsletter to stay informed about application!

- **Page des classes:** C'est la page qui contient la liste des classes de la faculté des sciences en fonction des filières.

FS SCHEDULER Home [Planning](#) [Log in](#)

Type a search

Classrooms Classes Teachers

Infos L2 Size:480 Get the scheduler	Infos L3 / Datascientist Size:480 Get the scheduler	Infos L1 Size:1000 Get the scheduler	Infos M1 / Securite Size:80 Get the scheduler	Infos M1 / Genie logiciel Size:80 Get the scheduler
Infos L3 / Genie logiciel Size:80 Get the scheduler	Infos L3 / Reseaux Size:80 Get the scheduler	Infos L3 / Securite Size:80 Get the scheduler	Infos M1 / Datascientist Size:80 Get the scheduler	

- **Page des enseignants** : C'est la page qui contient la liste des enseignant de la faculté des sciences en fonction de leur département d'appartenance.

FS SCHEDULER Home [Planning](#) [Log in](#)

Type a search

Classrooms Classes Teachers


[Informatics department](#)

NZEKON Armel Assistant Lecturer 	ESSOUMA Armand Professor 	MESSI Thomas Assistant Lecturer 	MONTHE Valery Senior lecturer 	BAYEM Narcisse Vaccataire 	TAPAMO Hyppolite Senior lecturer
MELATAGIA Paulin Senior lecturer 	Voltaire TCHAMBA Assistant Lecturer 	TSOPZE Norbert Professor 			

[Mathematics department](#)

- **Page d'enregistrement d'un cours** : C'est la page qui contient le formulaire d'enregistrement d'un cours.

FS SCHEDULER
Home
Planning
Personnalise
Logout



Classrooms
Classes
Teachers
Courses
Scheduler_form
Scheduler_planning

COURSE'S REGISTRATION FORM

Please complete the following form and submit it to register a course

Subject code * :

Libele * :


Credit * :

Main teacher * :

Second teacher :

- **Page d'enregistrement d'un salle** : C'est la page qui contient le formulaire d'enregistrement d'une salle de cours.

FS SCHEDULER
Home
Planning
Personnalise
Logout



Classrooms
Classes
Teachers
Courses
Scheduler_form
Scheduler_planning

CLASSROOM'S REGISTRATION FORM

Please complete the following form and submit it to register a classroom

Classroom name * :


Type :

Capacity :

* Required field

- **Page d'enregistrement d'un emploi de temps (manière classique) :** C'est la page qui contient le formulaire d'enregistrement d'un emploi de temps, par un formulaire proposé qui devra être dûment rempli par l'administrateur :

FS SCHEDULER
Home
Planning
Personnalise
Logout



Classrooms
Classes
Teachers
Courses
Scheduler_form
Scheduler_planning

Registration of a course's schedule for a class

Please fill the following form to register a course's schedule for a class

Course *:

Classroom *:


Teacher 1 *:

Teacher 2 :

Plage *:

- **Page d'enregistrement d'un enseignant :** C'est la page qui contient le formulaire d'enregistrement d'un enseignant dans la base de données:

FS SCHEDULER
Home
Planning
Personnalise
Logout



Classrooms
Classes
Teachers
Courses
Scheduler_form
Scheduler_planning

TEACHER'S REGISTRATION FORM

Please complete the following form and submit it to register a teacher

Registration number * :

Full name :

Email :

Profil picture :


Degree :

- **Page de téléchargement d'un emploi de temps** : C'est la page qui permet de télécharger ou imprimer l'emploi de temps crée :

FS SCHEDULER
Home
Planning
Log in

TIME TABLE OF Dr MELATAGIA Paulin

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
07h05 - 9h55							
10h05 - 12h55		A501 PHY3011 phy L3 (tronc commun)					
13h05 - 15h55							
16h05 - 18h55							
19h05 - 21h55							


Download

Conclusion

De la mise en évidence des besoins répondus par la base de données à son implémentation en SQL en passant par sa modélisation conceptuelle, nous avons pu mettre sur pied, dans le cadre pratique de l'unité d'enseignement intitulé Bases de données, une application capable de générer un emploi de temps pour la faculté des sciences de l'Université de Yaoundé 1. Il en ressort donc que l'utilisation de cette application permettra une bonne organisation de la faculté, ce qui favorisera une gestion judicieusement meilleure des infrastructures, du personnel et même des étudiants.