Systèmes d'Information Distribués

TDs

Mathieu Loiseau

GI-4-SID-S1

INSA de Lyon

septembre 2020

Table des matières

TD n° 1

Découverte de HTTP

Le présent TD requiert l'usage de la console de votre navigateur. Dans firefox comme chrome, elle est obtenue en pressant F12.

> La gestion du cache de Chrome (et Safari) fait qu'à part au premier chargement de la page, les questions 10 et 11 ne peuvent pas être traitée (à moins d'utiliser clic-droit sur le bouton d'actualisation et Empty cache and hard-reload).
> Le fonctionnement du *cache control* de Chrome fait qu'avec la configuration du serveur utilisé, il emploiera la stratégie du client 2 de la page suivante alors que l'on attend la stratégie du client 3. Cela rend la question 13 infaisable (au moins sur la durée d'un TD).

Nota Bene 1.1: Usage de Chrome pour le TD

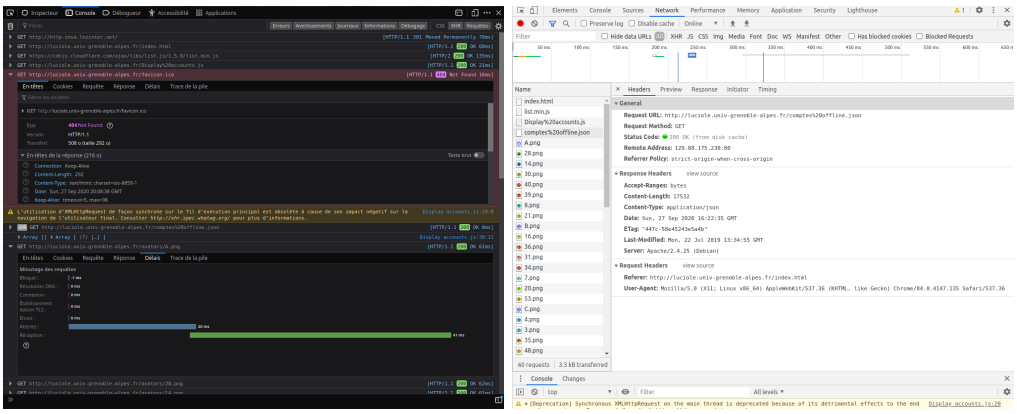

FIGURE 1.1 – Consoles Firefox & Chrome

Vous devez également être capables de vider le cache de votre navigateur pour pouvoir refaire certaines requêtes dans les bonnes conditions pour avoir les réponses attendues.

Les deux pages utilisées pour le TD sont :

— page1 : `http://http-insa.lezinter.net`

— page2 : `https://fr.wikipedia.org/w/index.php?title=Aide:Bac_%C3%A0_sable&action=edit`

1 GET

1. Ouvrez la console de votre navigateur puis chargez `page1`.
2. Regardez votre barre d'adresse. Que s'est-il passé ? *La ressource demandée a été déplacée. Le client suit automatiquement la redirection.*
3. Regardez le statut de la première requête. Que dit-elle de plus ? *La ressource demandée a été déplacée de manière permanente vers une autre adresse (code 301).*
4. Qu'indique le champ `location` de l'en-tête de réponse ? *Il contient l'adresse de la ressource demandée (adresse de redirection, utilisée dans la requête suivante).*
5. Quel est le statut de la seconde requête ? *200 (OK). La ressource a été trouvé et une représentation transmise.*
6. Quelle ressource est ciblée par la 3e requête ? *C'est un script JavaScript qui provient d'un autre serveur.*
7. Dans la console, comparez les versions de http de `luciole.univ-grenoble-alpes.fr` et de `cdn.cloudflare.com` ? *`luciole.univ-grenoble-alpes.fr` utilise la version 1.1, `cdn.cloudflare.com` la version 2.*
8. En observant les url et les temps de réponses (délais), expliquez l'autre différence de protocole entre ces deux serveur. *`luciole` utilise http quand `cloudflare` utilise https. https demande l'établissement d'une liaison TLS.*
9. Expliquer les temps de connexion de la requête suivante (`Display accounts.js`). *L'adresse IP du serveur est déjà connue, pas besoin d'une résolution DNS.*
10. Expliquer la réponse à la requête suivante (`favicon.ico`). *Le code de réponse 404 indique qu'aucune ressource (fichier, ici) `favicon.ico` n'a été trouvée à l'adresse indiquée.*
11. Regardez le `content-type` de la requête `favicon.ico`, que remarquez vous ? *La requête attendait une image, mais a obtenu du html (`text/html`), qui est la page par défaut proposée par le serveur quand la ressource n'est pas trouvée.*
12. Quels autres types de contenus ont été reçu lors des différentes requêtes effectuées ? Lesquels concernent des données, lesquels des programmes ?
    — *`text/html` (données)*
    — *`application/javascript` (programme)*
    — *`application/json` (données)*
    — *`image/png` (données)*
13. Recharger la page (sans vider le cache). Expliquer les temps de chargement (et code de réponse) des images. *Le client envoie dans sa requête le champ `If-Modified-Since` avec la date d'ajout de l'image à son cache. Comme le serveur lui renvoie le code 304 (`not modified`), il sait qu'il peut utiliser la version en cache et ainsi minimiser les temps de chargement.*

2 POST

14. Ouvrez la console de votre navigateur puis chargez `page2`.

15. Modifiez le texte et presser « publier ».
16. Quel est le code de réponse à la requête `POST` ? *302*
17. Expliquer la requête suivant le `POST`. *Le code 302 indique que la requête a été un succès. Le champ `location` indique vers quelle ressource orienter le client.*
18. Que contient le corps de la requête `POST` ? *L'intégralité du contenu de la ressource est envoyé et non uniquement la modification proposée.*

TD n° 2

Manipuler des données textuelles structurées en Python

Pour le présent TD, vous pouvez vous contenter d'utiliser un éditeur python en ligne comme `https://www.online-python.com/` .

1 Échauffement

Ce TD va traiter l'implémentation d'une partie d'un *client-lourd* d'une entreprise de dépannage. Le client recevra l'intégralité des sinistres non-traités et fournira à l'utilisateur des fonctionnalités pour sélectionner ceux qu'il traitera.

Nous allons coder ici des fonctions utilitaires que nous réutiliserons ultérieurement.

1. Chaque sinistre sera assorti de coordonnées GPS en degré sous forme de chaine de caractère (ex : `45.8841,4.8035`). Écrire une fonction `str2radians` qui prend en entrée cette chaine de caractères et renvoie une liste de `float` en radians. [1].
2. Écrire une fonction `distance` qui prend en entrée 2 chaines de caractères contenant des coordonnées GPS et renvoie grâce à la Formule de Haversine, la distance à vol d'oiseau entre ces deux points [2].

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2-\varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right)$$

Le code ci-dessous s'inspire de la solution proposée à ce problème.

```
1  def str2radians(coord):
2      from math import radians
3      res = []
4      #splits strings
5      for a in coord.split(","):
6          #adds each coordinate to a list
7          #after converting them to radians floats
8          res.append(radians(float(a)))
```

1. On pourra utiliser `math.radians`
2. `math` fournit également les fonctions `cos`, `sin`, `asin` & `sqrt`

```
9       return res
10
11  def distance(coord1, coord2):
12      from math import cos, sin, asin, sqrt
13     #initiates variables
14      lon1, lat1, lon2, lat2 = str2radians(coord1) + str2radians(coord2)
15      # formule de Haversine
16      dlon = lon2 - lon1
17      dlat = lat2 - lat1
18      a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
19      c = 2 * asin(sqrt(a))
20      r = 6371 #radius of the earth in km
21      return c * r
```

2 XML

Pour cette section nous allons utiliser le module python `xml.dom.minidom`[3], qui fournit une implémentation minimale du DOM. Celle-ci fournit une classe `Document` qui contient des objets `Node` dont certains (les balises du document initial) sont des `Element`s.

Pour cette partie nous utiliserons le document `incidents-FR.xml`.

3. Écrire une fonction `load_xml` qui prend un nom de fichier (`string`) en entrée et renvoie un objet `minidom.document`.[4]
4. Créer une fonction `get_locations` qui utilise `dom.element.getElementsByTagName` et renvoie la liste des lieu des sinistres..
5. Écrire une fonction `get_close_locations`, qui prend en entrée une liste de nœuds DOM avec des lieux, les coordonnées d'un point de référence et une distance maximale et qui renvoie une liste de nœuds DOM avec les lieux situés à moins de la distance maximale du point de référence.[5]
6. Écrire une fonction `get_close_incidents`, qui prend en entrée une liste de nœuds DOM avec des lieux, les coordonnées d'un point de référence et une distance maximale et qui renvoie une liste de nœuds DOM avec les **sinistres** situés à moins de la distance maximale du point de référence.[6]
7. Pour débuguer cette fonction, utiliser `getAttribute` et afficher l'`id` de chaque sinsitre sélectionné.
8. Écrire une fonction `assign_incident`, qui prend un document, un `id` de sinistre et le nom d'un opérateur et ajoute à l'élément concerné un nœud `operateur` contenant le nom de l'opérateur.[7]

3. On pourrait aussi utiliser ElementTree, qui est peut être plus simple, mais ne donne pas à voir le standard du DOM.

4. Cette fonction utilisera `minidom.parse`.

5. NB : le nœud textuel contenant l'information est le premier fils de l'élément `lieu` concerné. Utiliser `firstChild` & `nodeValue`.

6. NB : Le sinistre est le parent du nœud `lieu`. Pour le trouver on peut utiliser l'une des méthodes des objets `Node`.

7. Pour y parvenir, il va falloir ajouter au document un nœud `operateur` et un nœud textuel avec le nom de l'opérateur. Faire de ce nœud un fils du nœud `operateur` qui sera lui-même un enfant du nœud de le `sinistre` concerné. Cela requiert les méthodes suivantes
— `Node.appendChild()`;
— `Document.createElement()`;
— `Document.createTextNode()`;

9. Vous pouvez afficher le document résultant avec la méthode `toXML`.

```python
import xml.dom.minidom as minidom, utils
def load_xml(file_name):
    with open(file_name,"r") as input_xml:
        incidents = minidom.parse(input_xml)
    return incidents

def get_locations(document):
    return document.getElementsByTagName("lieu")

def get_close_locations(document, origin, max_dist):
    locations = get_locations(document)
    res = []
    for loc in locations:
        if(utils.distance(loc.firstChild.nodeValue, origin) < max_dist):
            res.append(loc)
    return res

def get_close_incidents(document, origin, max_dist):
    locations = get_close_locations(document, origin, max_dist)
    res = []
    for loc in locations:
        res.append(loc.parentNode)
        print(loc.parentNode.getAttribute("id"))
    return res

def assign_incident(document, id, operator):
    incidents = document.getElementsByTagName("sinistre")
    for inc in incidents:
        if inc.getAttribute("id") == id:
            operatorNode = document.createElement("operateur")
            operatorNode.appendChild(document.createTextNode(operator))
            inc.appendChild(operatorNode)
    return document
```

3 JSON

Pour cette section nous allons utiliser le module python `json` et le document `incidents-FR.json`.

Comme le module JSON ne fournit pas une API mais un mapping vers des structure de données python, la manipulation peut paraitre plus directe, nous allons donc reproduire le même travail que dans la section 2, mais en fournissant moins d'étapes.

10. Pour charger un document on utilisera la méthode `json.load`, qui fonctionne comme `minidom.parse`, mais renvoie une structure de données python (cf. diaporama). Créer la fonction `load_json(cheminDuFichierJson)`.
11. Écrire la fonction `get_close_incidents` de la question 6 en utilisant une structure JSON obtenue.
12. Écrire la fonction `assign_incidents` de la question 8 en utilisant une structure JSON.

```python
import json, utils
def load_json(file_name):
    with open(file_name,"r") as input_json:
        incidents = json.load(input_json)
```

```
    return incidents

def get_close_incidents(incidents, origin, max_dist):
   res = []
   for inc in incidents:
      if utils.distance(inc["lieu du sinistre"],origin) < max_dist:
         res.append(inc)
   return res

def assign_incident(incidents, id, operator):
   for inc in incidents:
      if inc["id"] == id:
         inc["operateur"] = operator
   return incidents
```

13. Vous pouvez utiliser `json.dumps(obj)`, pour voir le résultat.

```
import myXMLFR as myXML, myJSONFR as myJson, json
##XML version
language = "FR"
incx = myXML.load_xml(f"incidents-{language}.xml")
print(type(incx))
print(myXML.get_locations(incx))
print('myXML.get_close_incidents(incx, "45.8841,4.8035", 50)')
print(myXML.get_close_incidents(incx, "45.8841,4.8035", 50))
myXML.assign_incident(incx, "14", "John Doe")
print(incx.toxml())

#JSON version
incj = myJson.load_json(f"incidents-{language}.json")
print(myJson.get_close_incidents(incj, "45.8841,4.8035", 50))
#unlike the XML document,
#the JSON object can contain numbers and not just strings
myJson.assign_incident(incj, 14, "John Doe")
print(json.dumps(incj, indent=2))
```

TD n° 3

Des bases de données aux données structurées et *vice-versa*

Dans ce TD nous allons nous placer dans la situation d'un logiciel de gestion de discothèque/streaming audio, dont la base de données aurait la structure indiquée dans la figure 3.1.

Ce système, inspiré de ampache, doit permettre à une instance de partager son catalogue avec une autre instance. Ainsi, l'utilisateur connecté à une instance peut aussi bien lire des chansons de l'instance à laquelle il est connecté que chansons d'une autre instance liée. Chaque instance embarque un ensemble de catalogues qui ont un type `'local'` ou `'distant'` (cf. `` `catalog`.`catalog_type` ``), selon que les chansons sont hébergées par l'instance elle-même ou une instance connectée. Chaque instance répertorie donc toutes les chansons auxquelles elle donne accès, celles qu'elle héberge et celles d'autres instance qui lui partagent leur catalogue. Si la chanson est hébergée en local, elle sera lue directement par le client (navigateur de l'utilisateur) en suivant le chemin `` `catalog`.`path`/`song`.`file` ``, sinon celui-ci lira les données depuis l'instance distante (grâce à l'url stockée dans 'catalog'.'path' et à l'identifiant `` `song`.`file` ``).

Le problème auquel nous allons nous intéresser est l'échange de données entre les deux instances (et non le streaming audio).

Il est conseillé d'utiliser pour toutes les requêtes mySQL les mécanismes de gestion d'erreur de python en mettant toutes vos requêtes dans un bloc `try`.

```
1  #db is the handler of your mysql database
2  #obtained through an earlier call to the
3  #connect method
4  queries = ["SHOW TABLES;",
5             "DESC `distant-song`",
6             "SELECT * FROM `distant-songs`"]
7  try:
8        cursor = db.cursor()
9        for query in queries:
10          res = cursor.execute(query)
11 except mysql.connector.Error as e:
12    print(f"Error in ""{query}"": {e}")
```

Nota Bene 3.1: Erreurs en python

```
##catalog
+----------------+---------------------+------+-----+---------+----------------+
| Field          | Type                | Null | Key | Default | Extra          |
+----------------+---------------------+------+-----+---------+----------------+
| id             | int(11) unsigned    | NO   | PRI | NULL    | auto_increment |
| name           | varchar(128)        | YES  |     | NULL    |                |
| catalog_type   | varchar(128)        | YES  |     | NULL    |                |
| path           | varchar(255)        | YES  |     | NULL    | path or url    |
+----------------+---------------------+------+-----+---------+----------------+


## artist
+---------------+------------------+------+-----+---------+----------------+
| Field         | Type             | Null | Key | Default | Extra          |
+---------------+------------------+------+-----+---------+----------------+
| id            | int(11) unsigned | NO   | PRI | NULL    | auto_increment |
| name          | varchar(255)     | YES  |     | NULL    |                |
+---------------+------------------+------+-----+---------+----------------+


## album
+----------------+---------------------+------+-----+---------+----------------+
| Field          | Type                | Null | Key | Default | Extra          |
+----------------+---------------------+------+-----+---------+----------------+
| id             | int(11) unsigned    | NO   | PRI | NULL    | auto_increment |
| name           | varchar(255)        | YES  |     | NULL    |                |
| year           | int(4) unsigned     | NO   |     | 1984    |                |
| album_artist   | int(11) unsigned    | YES  |     | NULL    | artist.id      |
+----------------+---------------------+------+-----+---------+----------------+


## song
+------------+----------------------+------+-----+---------+--------------------+
| Field      | Type                 | Null | Key | Default | Extra              |
+------------+----------------------+------+-----+---------+--------------------+
| id         | int(11) unsigned     | NO   | PRI | NULL    | auto_increment     |
| file       | varchar(255)         | YES  |     | NULL    | path or distant id |
| catalog    | int(11) unsigned     | NO   |     | 0       | catalog.id         |
| album      | int(11) unsigned     | NO   | MUL | 0       | album.id           |
| year       | mediumint(4) unsigned| NO   |     | 0       |                    |
| artist     | int(11) unsigned     | NO   |     | 0       | artist.id          |
| number     | int(1) unsigned      | YES  | MUL | NULL    |                    |
| title      | varchar(255)         | YES  |     | NULL    |                    |
| play_count | int(11)              | YES  |     | 0       |                    |
+------------+----------------------+------+-----+---------+--------------------+
```

FIGURE 3.1 – Structure de la base de données relationnelle du système

1 Connexion à la base de données

Nous utiliserons le module `python-mysql.connector` en utilisant l'architecture décrite dans la figure 3.2.

> Un serveur a été mis à votre disposition (`lezinter.net`), pour ce TD. Un utilisateur unique est utilisé pour tout le groupe `insa-lyon` qui est aussi le nom de la base de données. Le mot de passe de l'utilisateur est le code associé à ce cours sur moodle et dans la plaquette de votre formation.

Nota Bene 3.2: Connexion au serveur mariaDB

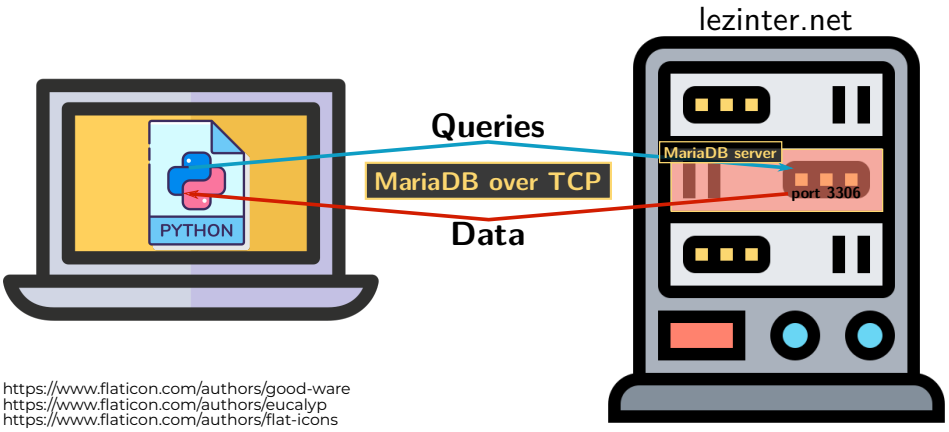

FIGURE 3.2 – Architecture pour le TD

1. Créer un fichier `dbconfig.py`, contenant 4 variables (`host`, `user`, `database`, `password`) instanciées avec les valeurs nécessaires pour se connecter au serveur (cf. NB 3.2).

```
1 host="lezinter.net"
2 user="insa-lyon"
3 password="I told you why it is not here"
4 database="insa-lyon"
```

Listing 3.1 – `dbconfig.py`

2. Créer une fonction `dbConnect`, qui utilise les variables de `dbconfig.py`, la méthode `mysql.connector.connect`, pour se connecter à la base de données. Cette fonction renvoie le handler de la base de données ou `False` si la connexion a échoué.
3. Tester la fonction précédente et refermer la connection avec la méthode `.close()`.

```
1 import dbconfig, mysql.connector, json
2 #connects to database
3 def dbConnect():
4     try:
5         db = mysql.connector.connect(
6         host=dbconfig.host,
```

```
7         user=dbconfig.user,
8         password=dbconfig.password,
9         database=dbconfig.database)
10    except mysql.connector.Error as e:
11        print(f"Error connecting to MariaDB Platform: {e}")
12        db = False
13    return db
```

Listing 3.2 – Connexion à la base de donnée

2 Initialisation de votre modèle

Chaque étudiant va se créer 4 tables, symbolisant le « modèle » d'une instance du système décrit, conformes aux spécifications de la figure 3.1.

Pour cela vous pouvez vous aider du fichier td3-your data structure.sql. Pensez-bien à modifier le nom des tables pour qu'ils soient uniques et associés à chaque étudiant (remplace `‘votrenom‘` par votre nom).

4. Créer une fonction `init` qui initialise votre base de données en créant les 4 tables associées à votre nom. Pour cela vous allez utiliser le « *handler* » de la base de donnée créé dans la section précédente. Il va vous permettre de créer un curseur avec la méthode `.cursor()`. C'est ce cuseur qui va vous permettre d'exécuter des requêtes SQL avec la méthode `.execute(requete)`.

```
1  #Creates a set of tables in a new database
2  def init(db):
3      res = True
4      try:
5          cursor = db.cursor()
6          queries = ["CREATE TABLE IF NOT EXISTS ‘ML-catalog‘(‘id‘ int(11)
               AUTO_INCREMENT PRIMARY KEY,‘name‘ varchar(128) NULL DEFAULT NULL,‘
               catalog_type‘ varchar(128) NULL DEFAULT NULL,‘path‘ varchar(255) NULL
               DEFAULT NULL );",
7              "CREATE TABLE IF NOT EXISTS ‘ML-artist‘ (‘id‘ int(11) AUTO_INCREMENT
                   PRIMARY KEY,‘name‘ varchar(255) NULL DEFAULT NULL );",
8              "CREATE TABLE IF NOT EXISTS ‘ML-album‘ (‘id‘ int(11) AUTO_INCREMENT
                   PRIMARY KEY,‘name‘ varchar(255) NULL DEFAULT NULL,‘year‘ int(4)
                   unsigned NOT NULL DEFAULT 1984,‘album_artist‘ int(11) unsigned NULL
                   DEFAULT NULL);",
9              "CREATE TABLE IF NOT EXISTS ‘ML-song‘ (‘id‘ int(11) AUTO_INCREMENT
                   PRIMARY KEY,‘file‘ varchar(255) NULL DEFAULT NULL,‘catalog‘ int(11)
                   unsigned NOT NULL DEFAULT 0,‘album‘ int(11) unsigned NOT NULL DEFAULT
                    0,‘year‘ mediumint(4) unsigned NOT NULL DEFAULT 0,‘artist‘ int(11)
                   unsigned NOT NULL DEFAULT 0,‘number‘ int(1) unsigned NULL DEFAULT
                   NULL,‘title‘ varchar(255) NULL DEFAULT NULL,‘play_count‘ int(11) NULL
                    DEFAULT 0);"]
10         for query in queries:
11             cursor.execute(query)
12     except mysql.connector.Error as e:
13         print(f"Error init in \"{query}\": {e}")
14         res = False
15     return res
```

Listing 3.3 – fonction `init`

5. Ajouter dans votre table `votrenom-catalog` un unique catalogue local, en exécutant une requête `INSERT`. Vos insertions ne seront effective qu'après avoir appelé la méthode `.commit()` sur le handler de votre base de données.
6. Créer un album de votre choix dans votre base de données (en ajoutant donc les chansons et le(s) artiste(s) associés).
7. Pour vérifier que tout s'est passé comme vous l'attendiez, créez une fonction `show_table_content(cursor, table)`, qui prend un curseur sur une base de données et le nom d'une table et affiche son contenu. Appelez cette fonction sur vos 4 tables. Cette fonction utilisera sur le curseur la méthode `fetchall()`, qui récupère tous les résultats et les met dans une `liste` de `tuple`s. [1]

```python
#shows the content of a table
def show_table_content(db, tablename):
    res = True
    cursor = db.cursor()
    query = "SELECT * FROM `" + tablename + "` ;"
    try:
        cursor.execute(query);
        myresult = cursor.fetchall()
        for x in myresult:
          print(x)
    except mysql.connector.Error as e:
        print(f"Error show_table_content in \"{query}\": {e}")
        res = False
    return res
```

Listing 3.4 – `show_table_content`

3 Contrôleur

Dans cette section nous allons manipuler les données.

3.1 Opération basique

8. Créer une fonction `add_1_to_play_count(db, song_id)` qui ajoute 1 au nombre de lectures de la chanson d'id `song_id`. Cette fonction doit manipuler exclusivement vos tables.

```python
#adds one to the number of plays of a given song
def add_1_to_play_count(db, song_id, prefix="distant"):
    res = True
    try:
        cursor = db.cursor()
        #you must use yourname-song not the global one
        query = "UPDATE `{prefix}-song` SET `play_count` = `play_count` + 1 WHERE `
            id` = '" + str(song_id) + "';"
        cursor.execute(query)
        db.commit()
    except mysql.connector.Error as e:
        print(f"Error add_1_to_play_count in \"{query}\": {e}")
        res = False
```

1. Vous pouvez aussi appeler cette fonction sur la table `distant-song`.

```
13     return res
```

Listing 3.5 – Compter les lecturs

3.2 Export

9. Définissez un formalisme XML ou JSON apte à contenir les chansons d'un catalogue local.
10. Créer une fonction `data_export(db, catalog_id)` qui utilise ce formalisme pour exporter un catalogue de votre instance.

```
1  #exports the content of a local catalog returns false on Error
2  #or if the catalog does not exist or if it is not a local catalog
3  def data_export(db, catalog_id, prefix="distant", to_json=True):
4      res = dict()
5      try:
6          #get all songs
7          query = f"SELECT `{prefix}-song`.`id`, `{prefix}-album`.`name`, `{prefix}-
               song`.`year` as `year`, `{prefix}-album`.`year` as `Ayear`, `artist`.`
               name`, `album-artist`.`name`, `{prefix}-song`.`number`, `{prefix}-song
               `.`title` FROM `{prefix}-catalog`, `{prefix}-song`, `{prefix}-album`, `{
               prefix}-artist` as `artist`, `{prefix}-artist` as `album-artist` WHERE
               `{prefix}-catalog`.`id`='{catalog_id}' AND `{prefix}-catalog`.`
               catalog_type`='local' AND `{prefix}-song`.`catalog`='{catalog_id}' AND `
               artist`.`id`=`{prefix}-song`.`artist` AND `{prefix}-song`.`album`=`{
               prefix}-album`.`id` AND `{prefix}-album`.`album_artist` = `album-artist
               `.`id` ORDER BY `artist`, `Ayear`, `{prefix}-album`.`name`, `{prefix}-
               song`.`number`;"
8          cursor = db.cursor()
9          cursor.execute(query)
10         results = cursor.fetchall()
11         if(len(results) == 0):
12             print("No local catalog with id "+ str(catalog_id))
13             res = False
14         else:
15             for line in results:
16                 if not line[5] in res:
17                     res[line[4]] = dict()
18                 if not line[1] in res[line[5]]:
19                     res[line[5]][line[1]]=dict(year=line[3],songs=[])
20                 res[line[5]][line[1]]["songs"].append(dict(
21                     id=line[0],
22                     number=line[6],
23                     artist=line[4],
24                     title=line[7],
25                     year=line[2]
26                 ))
27     except mysql.connector.Error as e:
28         print(f"Error export in \"{query}\": {e}")
29         res = False
30     if(to_json and res != False):
31         res = json.dumps(res, indent=3)
32     return res
```

Listing 3.6 – `export_data`

3.3 Import

11. Créez une fonction `add_artist(cursor, name)`, qui ajoute un artiste à la table concernée (uniquement si l'artiste n'existe pas encore).

```
#inserts a new artist in the database (if necessary) and returns their id
def add_artist(cursor, artist_name, prefix="distant"):
    try:
    #in case SQL special characters are in the strings (e.g. ') we escape them
    #with dbConection.converter.escape() or cursor._connection.converter.escape()
        artist_name = cursor._connection.converter.escape(artist_name)
        query = f"SELECT `id` FROM `{prefix}-artist` WHERE `name`='{artist_name}';"
        cursor.execute(query)
        results = cursor.fetchall()
        #get album artist id
        if(len(results)==0):
            #create artist
            query = f"INSERT INTO `{prefix}-artist` (`name`) VALUES ('{artist_name}')
                ;"
            cursor.execute(query)
            artist_id=cursor.lastrowid
        else:
            artist_id = results[0][0]
    except mysql.connector.Error as e:
        print(f"Error add_artist in \"{query}\": {e}")
        artist_id = False
    return artist_id
```

Listing 3.7 – `add_artist`

12. Créez une fonction `add_album(cursor, name, year, artist_id)`, qui ajoute un album à la table concernée (uniquement s'il n'existe pas encore).

```
#inserts a new album in the database (if necessary) and returns its id
def add_album(cursor, album_title, album_year, artist_id, prefix="distant"):
    try:
        #check if album exists
        album_title = cursor._connection.converter.escape(album_title)
        query = f"SELECT `id` FROM `{prefix}-album` WHERE `name`='{album_title}' AND
            `album_artist`='{artist_id}';"
        cursor.execute(query)
        results=cursor.fetchall()
        if(len(results) == 0):
            #create album
            query = f"INSERT INTO `{prefix}-album` (`name`,`year`,`album_artist`)
                VALUES ('{album_title}','{album_year}','{artist_id}');"
            cursor.execute(query)
            album_id=cursor.lastrowid
        else:
            album_id = results[0][0]
    except mysql.connector.Error as e:
        print(f"Error add_album in \"{query}\": {e}")
        album_id = False
    return album_id
```

Listing 3.8 – `add_album`

13. Créez une fonction `add_song(cursor, song_id, catalog_id, album_id, artist_id, number, title, year)`, qui ajoute une chanson à la table concernée (uniquement si

elle n'existe pas encore). Attention, si vous avez bien créé vos table avec les requêtes fournies, c'est le SGBD qui vous enverra une erreur si la chanson existe déjà [2].

```python
#inserts a new song in the database (if necessary) and returns its id
def add_song(cursor, song_id, catalog_id, album_id, artist_id, number, title, year
    , prefix="distant"):
   res = 1
   try:
      query = f"INSERT INTO `{prefix}-song` (`file`, `catalog`, `album`, `year`, `
          artist`, `number`, `title`) VALUES ('{song_id}', '{catalog_id}', '{
          album_id}', '{year}', '{artist_id}', '{number}', '{cursor._connection.
          converter.escape(title)}');"
      cursor.execute(query)
      #due to the table structure, a song that is already present (same album and
          track number)
      #will not be added. We can handle the error and go on with our treatment
   except mysql.connector.Error as e:
      print(f"{query} yielded the following error: {e}\n{title} was not added")
      res = 0
   return res
```

Listing 3.9 – `add_song`

14. Utiliser les fonctions précédentes pour créer une fonction `data_import(db, catalog_id)` qui utilise ce formalisme pour importer un catalogue vers votre instance. Attention, cette fonction ne doit importer que les chansons qui ne sont pas présentes sur votre instance. Cette fonction ne doit exécuter les changements que si tout s'est bien passé (`db.commit()` en fin de fonction).
15. Adaptez votre fonction d'export pour obtenir un fichier de données du catalogue 1 des tables `` `distant-*` `` et les importer dans un catalogue distant de votre base de données avec `data_import`.

```python
#imports the content of a json object into
#a new distant catalog "catalog_name" of url "catalog_url"
#the songs already present in another catalog are not added
def data_import(db, catalog_name, catalog_url, data, prefix="distant"):
   count=0
   try:
      cursor = db.cursor()
      #Create catalog
      query = f"INSERT INTO `{prefix}-catalog` (`name`, `catalog_type`, `path`)
          VALUES ('{catalog_name}', 'distant', '{catalog_url}');"
      cursor.execute(query)
      catalog_id=cursor.lastrowid
      #process data
      data = json.loads(data)
      for artist, discography in data.items():
         #get album artist id
         artist_id=add_artist(cursor, artist)
         if(artist_id):
            #process discography
            for album_title, album_data in discography.items():
               album_id = add_album(cursor, album_title, album_data['year'],
                   artist_id)
               #process songs
```

2. Si vous utilisez habilement les exceptions vous gagnerez du temps mais vous pouvez vous en sortir sans.

```
22                if(album_id):
23                   previous_artist=False
24                   for song in album_data["songs"]:
25                      if(song["artist"] != previous_artist):
26                         artist_id=add_artist(cursor, song["artist"])
27                         previous_artist=song["artist"]
28                      count += add_song(cursor, song['id'], catalog_id, album_id,
                            artist_id, song['number'], song['title'], song['year'])
29          except mysql.connector.Error as e:
30             print(f"Error in import: {e}")
31             count=False
32          if count > 0:
33             #if all the previous queries worked
34             #and at least 1 track was added,
35             #we "commit" the changes (= modify the content of the database)
36             #otherwise nothing happens to the database
37             db.commit()
38          return count
```

Listing 3.10 – `data_import`

TD n° 4

Projet — simulation d'application de traçage « Covid » décentralisée

Ce projet vise à donner un aperçu, très simplifié, d'une application de traçage décentralisée. Ceci va nous permettre d'aborder plusieurs notions :

- Architecture client-serveur ;
- APIs RESTful.

Il sera également un prétexte à :

- Gérer votre propre serveur depuis chez vous et le rendre accessible ;
- Utiliser une méthode de développement "Agile" ;
- Gérer des versions de document avec git ;
- Orchestrer tout cela avec gitlab.

Il s'agit ici de comprendre une solution technique (dp^{3}t), mais ce sujet n'exprime pas d'opinion sur sa pertinence [1]...

dp^{3}t est décrit dans la figure 4.1 p. 20 sous forme de BD.

1 Configurer l'environnement

1.1 Environnement de développement

À part la récupération du code de base toute cette partie n'est pas *nécessaire* à la réalisation du projet. Cependant, elle vous donnera à voir des méthodes de gestion de projet qui peuvent vous être utiles. De plus l'usage de git avec un dépôt distant facilitera également le partage de votre code.

1. Créer un compte sur framagit.org
2. Constituez un groupe de 3 à 4 étudiants.
3. L'**un des étudiants du groupe** crée un « fork » du projet suivant et invite les autres en tant que *maintainer*. :

1. Pour cela, vous pouvez consulter `https://risques-tracage.fr/docs/risques-tracage.pdf`

FIGURE 4.1 – Explication des principes de dp^{3}t
https://github.com/DP-3T/documents/tree/master/public_engagement/cartoon

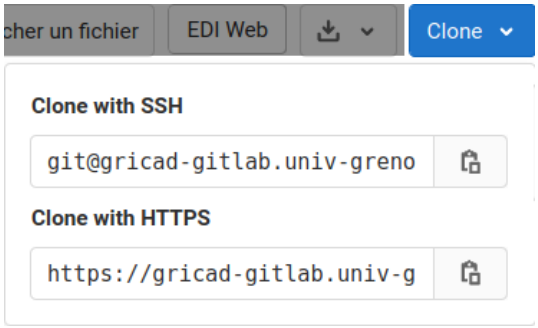

FIGURE 4.2 – Pour obtenir l'adresse du dépôt à cloner, presser `clone`, puis le bouton associé au protocole de votre choix.

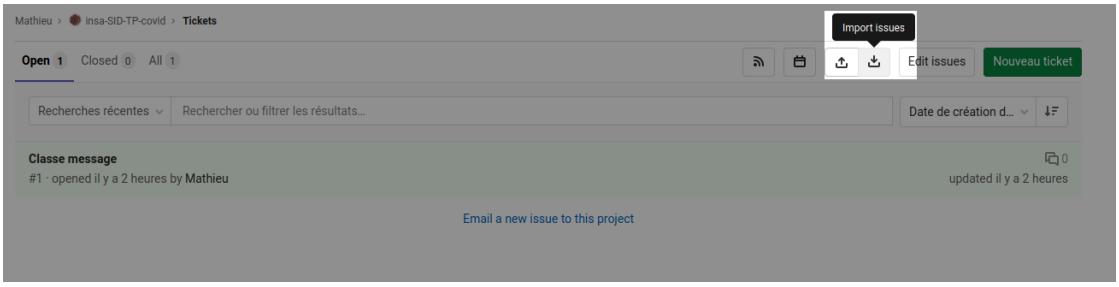

FIGURE 4.3 – `tickets→import`

`https://framagit.org/lzbk/insa-sid-tp-covid`
Une méthode alternative est de créer un groupe gitlab et de *forker* le projet pour le groupe.

4. Installer git sur votre ordinateur depuis `http://git-scm.com/downloads`.
5. Configurer git avec vos nom et adresse e-mail.
   — Soit par la ligne de commande (préferez `git-bash` sous MS windows)
   ```
   git config --global user.name "<nom>"
   git config --global user.email "<email>"
   ```
   — soit avec une interface graphique de votre choix.
   `https://git-scm.com/downloads/guis`
6. Cloner votre projet distant sur votre machine (Interface graphique ou `git clone <dépôt distant>`). Cf. fig.4.2.
7. Importer la liste des tickets à votre dépôt (`issues.csv`). Cf. fig.4.3.

1.2 Réseau

Pour l'essentiel des développements vous allez travailler localement sur votre machine (en utilisant l'adresse IP `127.0.0.1` ou `localhost`), mais à terme votre ordinateur devra être accessible à d'autres.

Pour cela il vous faudra configurer votre box pour que les communication entrante sur le port 5000 soient envoyées à votre ordinateur.

Pour tester cela, nous allons utiliser le code que vous avez téléchargé sur votre machine. Celui-ci nécessite les librairies `vitualenv` et `flask`.

8. Suivez les consignes du `README.md`, pour installer la version nue du projet.
   `https://framagit.org/lzbk/insa-sid-tp-covid#mise-en-place-de-lenvironnement`
9. Lancer l'application (fichier `app.py`).
10. Connectez-vous avec votre navigateur à l'adresse `http://localhost:5000` et assurez-vous qu'un message s'affiche.
11. Transférez le port 5000 à votre ordinateur :
    `https://www.somfypro.fr/assistance/faq/alarme/-/asset_publisher/QyK3/content/comment-ouvrir-les-ports-sur-une-box-internet` *Le tutoriel ne transmet pas le port 5000, mais un autre, il faut adapter la configuration.*
12. Vérifiez que la redirection de port fonctionne en demandant à l'un des membres de votre groupe de se connecter à votre port 5000 avec son navigateur (vous pouvez obtenir votre adresse ip à lui transmettre en vous connectant à un site type `ipchicken.com`).

2 Le projet

À la différence d'un TP un projet demande que vous preniez le temps d'analyser le problème qui vous est posé et de le décomposer en sous problèmes de taille gérables. Le projet est décomposé en trois phases

1. Installation, création de votre environnement de travail, test de celui-ci (phase qui suit normalement l'analyse des besoins) ;
2. Analyse des besoins ;
3. Implémentation.

En « agilité », lors de la première séance, la première phase doit être exécutée. Ensuite les deux autres phases sont abordées sur les deux séances :

— Lors de la première séance nous nous focaliserons sur les besoins liées aux fonctions de gestion des messages ;
— Lors de la deuxième séance nous nous focaliserons sur les échanges réseau (définition des routes) et le test.

> *Une* liste de méthodes utiles pour la réalisation du projet est disponible sur la branche `master` du dépôt. Il y a d'autres manières de résoudre ce problème, mais cela vous offrira des pistes.

Nota Bene 4.1: Spécifications

3 Évaluation

Le projet étant ambitieux par rapport aux circonstances, il a été décidé de modifier les modalités d'évaluation :

— *Une* correction du projet a été mise en ligne sur la branche `correction` du dépôt ;

— les étudiants ayant déjà avancé sur leur projet peuvent s'en servir (notamment les routes pour l'interface de test de l'application), ils expliqueront comment fonctionne *leur* version et ce qu'ils ont récupéré de la correction ;
— les autres étudiants fourniront un commentaire détaillé de la version corrigée ;
— l'évaluation tiendra compte de l'avancement des uns et des difficultés des autres. . .
— tous les contenus seront rendu par le biais de votre dépôt git avant le *29 novembre*.

Annexe A

(F)AQ

Ici, une liste de questions qui m'ont été posées au moins une fois avec une réponse.

1 Réseaux

1. Au niveau du tableau représentant les différentes couches TCP/IP diapo 21, je ne comprends pas bien si cela implique qu'une application type http est forcément liée à un transport de type TCP qui lui-même est lié à un réseau IP via une liaison ethernet cuivrée ? (Fonctionnement par colonne) et si oui, comment interpréter les deux dernières colonnes du tableau ? *La réponse est non. Si http fonctionne toujours avec TCP, le tableau ne représente pas un lien direct. Sur le principe, la notion de couches veut dire que les couches supérieures s'appuient sur les couches inférieures et que sans câbles en cuivres, ondes wifi ou fibre optique, pas de tcp. Mais les technologies des couches supérieures peuvent avoir déléguer à plusieurs technologies des couches inférieures.*
2. Je ne comprends pas la notion de sous-réseaux, j'ai trouvé sur internet qu'un masque de sous réseau était « un séparateur entre la partie réseau (bits à 1) et la partie machine (bits à 0) d'une adresse IP » mais ça ne m'éclaire guère plus. *Pour comprendre cette phrase il faut comprendre la page. Un sous-réseau est un réseau physique la plupart du temps. Le réseau local de votre logement, le réseau de l'INSA, mais cela peut avoir une granularité plus grande (un sous-réseau peut contenir lui même des sous-réseaux : on peut considérer le réseau de l'INSA comme un sous-réseau de RENATER, lui-même sous-réseau de l'internet). La notion de masque de sous-réseau relève de la notion d'adresse IP. Dans une adresse courrier, on va séparer la rue de la ville en allant à la ligne, dans une adresse IP, selon la taille du sous-réseau, on va avoir besoin de plus ou moins de bits pour identifier le sous-réseau. D'où cette notion de masque.*

   Pour revenir à ta source internet, la partie machine représente les bits de l'adresse IP qui désignent (numérotent, au final) les machines d'un sous-réseau. La partie sous-réseau identifie les bits de l'adresse IP qui désigne le sous-réseau.
3. Au niveau du TCP, je comprends la notion de connexion mais pas celle de segmentation, qu'est-ce qu'elle implique ? *La segmentation implique de pouvoir envoyer de gros messages, en les découpant en segments. Cependant chaque segment peut se perdre... De ce fait, il faut des mécanismes pour vérifier que tous les segments sont arrivés et les*

redemander le cas échéant. Les segments peuvent aussi se mélanger, il faut donc que le protocole définisse des mécanismes pour remettre dans l'ordre des segments arrivés dans le désordre...

2 HTTP

4. Concernant l'URL je ne comprends pas bien sur la diapo 41, quelle est la différence entre le nom de domaine relié à la Machine et le port relié à l'application. Je pensais que les ports étaient intrinsèquement liés à une machine justement. *Le port est comme la boite aux lettre d'un immeuble. Si l'adresse IP mène à l'immeuble (la machine), le port mène à l'appartement (l'application qui va répondre à la requête). Donc, le port est lié à la machine dans le sens où toutes les machines n'écoutent pas sur tous les ports. Cependant, la plupart des machines qui ont un serveur Web écoutent sur le port 80 (et c'est là que s'arrête l'analogie avec l'immeuble et l'appartement...).*
5. Qu'est que « .php » dans ce même exemple ? Est-ce un langage de programmation au même titre qu'HTML ? *Oui... dans le sens où c'est une ressource que le serveur mettra à la disposition du client grace à son identification par l'url. Et non... Car html est un langage de structuration de document et php est un langage de script qui est interprété par le serveur pour générer du html ou du xml, ou du json etc. C'est là toute l'importance de l'en-tête de la requête http. Le client demande à voir un fichier .php. Le serveur exécute le script, qui génère une page html. Celle-ci est renvoyée au client, qui aura comme type* `text/html` *et pourra ainsi l'interpréter comme il faut. Ça se clarifiera après la dernière séance de TP où votre serveur renverra du JSON ou du XML...*
6. Toujours dans cet exemple, pourquoi affecte-t-on des valeurs (à a et à b ici) au sein même de l'adresse URL s'il ne s'agit que d'identifier une ressource ? *`a` et `b` sont, dans ce cas, des variables qui serviront au script php interprété par le serveur à personnaliser la page html envoyée au client.*
7. Dans quel cas on demande une représentation de la ressource (diapo 45) ? *On demande toujours une représentation de la ressource. La ressource reste sur le serveur (ou dans le monde réel)*
8. De ce que j'ai compris HTTP est un protocole et HTML un langage. Mais en recherchant sur internet des exemples, je suis tombée sur un GET du type `GET [/ http://www.commentcamarche.net] HTTP/1.0` et un autre du type `GET /repertoire/page.html HTTP/1.1` Dans l'un on spécifie le langage HTML dans l'autre non, pourquoi ? *En fait, on ne spécifie pas le langage, mais l'extension du fichier. Quand une requête http cible un dossier (comme dans la première requête), le serveur cherchera (par convention) par défaut un fichier* `index.html` *ou* `index.php` *(si le serveur Web a un module php)*
9. Pour la deuxième partie sur POST je ne trouve pas comme vous dans le code de réponse à la requête. Je trouve 204 et non pas 302. *204 indique une mise en cache ou une non modification de la ressource. Il faut bien faire une modification avant d'effectuer la requête.*

10. D'ailleurs, je ne comprends pas pourquoi le code 302 indique que la requête a été un succès. Est ce qu'il n'indique pas plutôt un redirection ? *En tant que réponse à une requête POST, 302 indique que la requête a été acceptée et qu'il faudra ensuite effectuer un GET sur une autre "location".* [1]

3 Échange de données

11. Je ne comprends pas le lien dans le cours entre le langage ASCII et les applications type json et xml. Quel est-il ? *ASCII n'est pas un langage, mais un encodage... C'est un peu comme les couches « Internet ». Un fichier texte doit avoir un encodage (ASCII, Windows-Latin-1, Iso 8859-1, utf-8) qui fait le lien entre la valeur d'un octet et un glyphe (une lettre, en gros). Mais, on peut écrire un texte en français ou l'écrire en html, c'est à dire en organisant nos caractères pour suivre les règles syntaxiques du html (ou du python, du xml, du json ou php, etc.)*

12. Dans la diapo 52, Elément a un conteneur qui contient plusieurs attributs et je ne comprends pas ce qui peut être ajouté dans « Contenu de l'élément » autre que des attributs qui seraient situés dans le premier « <> ». Idem pou élément vide, je ne saisis pas son utilité. *Je donne des exemples de chaque dans le TP n°2 dans ma représentation XML (cf. fig A.1).*

```
1     <sinistre id="5">
2         <ouverture>2020-09-30</ouverture>
3         <lieu>45.1841,5.7154</lieu>
4         <tractage />
5     </sinistre>
```

Listing A.1 – Attributs, contenu et marqueurs

13. Je ne comprends pas ce que fait « Déclaration » *La déclaration indique au logiciel qui va ouvrir le document qu'il s'agit d'un document qui suit la syntaxe XML (version 1.0) et qu'il devra être interprété comme tel.*

14. Xml est-il un langage au même titre qu'HTML ? Quel est sa valeur ajoutée par rapport à ce dernier ? *HTML sert à structurer des documents, XML est générique et sert à structurer tout ce qu'on veut (on définit nos propres balises).*

1. C'est un usage abusif qui vise à fonctionner avec les navigateurs qui n'implémentent pas la norme HTTP 1.1. « Many web browsers implemented this code in a manner that violated this standard, changing the request type of the new request to GET, regardless of the type employed in the original request (e.g. POST).[1] For this reason, HTTP/1.1 (RFC 2616) added the new status codes 303 and 307 to disambiguate between the two behaviours, with 303 mandating the change of request type to GET, and 307 preserving the request type as originally sent. Despite the greater clarity provided by this disambiguation, the 302 code is still employed in web frameworks to preserve compatibility with browsers that do not implement the HTTP/1.1 specification. »`https://en.wikipedia.org/wiki/HTTP_302`

4 Bases de données

15. Je ne saisis pas la nuance entre PRIMARY KEY et UNIQUE. *Il peut y avoir plusieurs colonnes contenant des valeurs uniques, mais qu'une seule clé primaire. De plus tous les enregistrements doivent avoir une valeur associée à une clé primaire, mais la valeur unique n'est pas obligatoire.*

16. Que fait précisemment le `cursor._connection.converter.escape(artist_name)` Pourquoi ne pas juste mettre `artist_name` ? *Si vous mettez `artist_name` et que l'artiste s'appelle `Booker T. & the M.G.'s`, vous allez vous retrouver avec une requête du type*
`INSERT INTO 'artist' ('name')VALUES ('Booker T. & the M.G.'s');` ce qui donnera une erreur le fait d'échapper automatiquement les caractères spéciaux générera :
`INSERT INTO 'artist' ('name')VALUES ('Booker T. & the M.G.\'s');`

17. Toujours sur ce code ci-contre, pourquoi ajouter un « prefix » devant artist ? *Ça permet de faire une correction générique qui marche avec toutes vos tables, il suffit de remplacer la variable `prefix` par le préfixe de vos tables et les fonctions marcheront.*

18. Pourquoi j'ai un unsigned pour 'artist' dans la création de la table ML-song par exemple et non pour 'id' dans la création de la table ML-artist ? *C'est une erreur... On gache des octets, puisque la colonne pourra accueillir une valeur négative que l'on utilisera jamais (ex : `TINYINT` pourra coder un nombre entre -128 et 127 alors que `TINYINT UNSIGNED` codera un nombre entre 0 et 255).*

19. Aussi, on spécifie « autoincrement » qui se trouve dans la colonne extra dans le code SQL mais pas « path or url » dans ML-catalog ni « artist.id » dans ML-album, pourquoi ? *`AUTO_INCREMENT` est une vraie instruction SQL, alors que `path or url` est un commentaire pour indiquer le type de chaine de caractère attendu.*

20. Que fait la méthode « commit » que l'on appelle sur le handler dans la question 5 du TD3 ? *La méthode `.commit` fait la même chose pour une transaction de base de donnée que ce qu'elle fait pour un historique « git » : elle ancre les modifications faites par les requêtes précédentes. Sinon les requêtes sont uniquement simulées et aucun changement n'est fait aux tables concernées.*
Voir la Documentation de python mysql connector

21. Pour la question sur data_export, pourquoi n'ai-je ici pas de « {prefix} » devant artist surligné dans l'image ci-contre alors que j'en ai partout ailleurs ? *Il s'agit ici d'une jointure où deux champs distincts (`'song'.'artist'` et `'album'.'album_artist'`) ont tous deux la même clé étrangère. J'ai donc dû avoir recours à deux occurrences de la même table. J'ai donné un nom différent à chacune de ces occurrences :*
`FROM '{prefix}-artist' as 'artist', '{prefix}-artist' as 'album_artist'`. Dans le reste de la requête je manipule directement ces instance avec le nom que je leur ai donné (`'artist'` et `'album_artist'`).

22. La méthode « fetchall() » récupère tout mais elle récupère quoi au juste ? Par exemple ci-contre on a results= cursor.fetchall(). *Elle récupère tous les résultats de la dernière requête (Cf. les nombreux tutoriels sur le sujet).*

23. Dernière question sur ce que fait le « cursor.lastrowid » dans la fonction add_album concrètement ? *Concrètement, cet attribut renvoie la valeur du champ `'album'.'id'` (le champ `AUTO_INCREMENT`) lors de la dernière insertion (et donc lors du dernier incrément de ce champ). Si on a ajouté un nouvel album dont l'id a automatiquement été fixé à 5, il vaudra 5.*
Cf. documentation python mysql connector.

5 Python

24. `print(f"Error in \"{ query }\": {e}")` Je ne saisis pas pourquoi il y a ce « f » avant le message à afficher ? *Le « f » est une manière pratique de* formatter *les chaines de caractères depuis python 3.6. La* f-string *ci-dessus est équivalente à* `print("Error in \""+ query + "\": "+ e)`.

6 Projet

25. `if __name__ == "__main__": #NE COMPRENDS PAS`
Ce bloc ne sera exécuté que si le fichier python qui le contient est exécuté directement. Cela permet de mettre du code de test qui ne soit pas exécuté lors d'un `import`.

26. `#a method to convert the object to string data #A QUOI SERT CETTE METHODE ??`
`def __str__(self):`
C'est la méthode appelée par python quand il essaie de convertir une variable en string. ous pouvez l'appeler explicitement `str(monObjet)`. *C'est pratique pour débugger.*

27.
```
#export/import
    def m_export(self):
    #CREER UN FICHIER JSON DE MESSAGES ??
    pass
    def m_import(self, msg):
    #ON VEUT IMPORTER QUOI??
    pass
```

Votre application doit se rappeler des messages qu'elle a dit et entendus même quand on l'éteint pour la nuit. Il faut pouvoir les stocker. Si ce sont les méthodes `MessageCatalog.c_export` *et* `MessageCatalog.c_import` *qui vont gérer le stockage (dans une BD, un fichier XML, un fichier JSON ou autre), il faut que chaque message puisse individuellement être exporté et réimporté. Du coup, au moment d'écrire* `MessageCatalog.c_export` *et* `MessageCatalog.c_import` *vous serez content d'avoir déjà écrit* `Message.m_export` *et* `Message.m_import`

Dans mon implémentation, je stocke les messages dans un fichier json que je réimporte à chaque lancement du programme. Dans mon test, le fichier concerné est `test.json`. *S'il n'existe pas encore, il est créé et s'il existe il est mis à jour. Encore une fois, le format json n'est pas obligatoire (BD, xml, txt, etc.) par contre, il faut pouvoir conserver les données.*

28.
```
    self.msg_type= #??
    #...
    mySecondMessage = Message(Message.generate(), Message.MSG_THEY) #CE
        N'EST PAS LE BON CONSTRUCTEUR!!!???
    #...
    #Check if message string is in a category
    def check_msg(self, msg_str, msg_type=Message.MSG_THEY): #QU EST CE
        QUE C EST MSG_THEY? IL EST DECLARE NUL PART. ET CA NE COMPILE
        PAS ICI NON PLUS.
```

Votre système doit gérer trois types de messages. Les messages que vous avez dits, les messages que vous avez entendus et les messages qui ont été dits par des malades du covid. Dans ma proposition, j'utilise des constantes liées à la classe message (`Message.MSG_ISAID`, `Message.MSG_IHEARD` & `Message.MSG_THEY`*). Vous n'êtes pas obligés d'utiliser la*

même implémentation (j'aurais donc dû supprimer cette valeur par défaut dans les paramètres), mais il faut pouvoir distinguer ces 3 types de messages dans votre catalogue.
`MessageCatalog.check_msg`permet de dire si dans la liste des messages d'un `msg_type` donné (ex : messages dits par les malades du covid), vous trouvez le message dont le contenu est `msg_str`.

29. Au final, ça sert à quoi la redirection de ports ? *Comme on l'a vu les adresses IP (v4, qui sont encore largement utilisées) sont chères (dans le sens où la combinatoire commence à être limitée). Votre fournisseur d'accès vous vend une unique adresse IP associée à votre box-modem-routeur. Du point de vue de l'extérieur (WAN) tous les périphériques connectés ont la même adresse IP (celle de votre box).*
Cependant de votre côté de la box (LAN) il faut pouvoir acheminer les paquets jusqu'à chaque terminal. Ceux-ci sont donc identifiés avec une adresse IP (interne), qui n'est connue que des périphériques connectés à la box. En général, cet adresse est en `192.168.1.*`. Pour tous les usages où vous êtes client, une fois que le circuit virtuel a été initialisé tout se passe bien. Mais si vous êtes serveur, votre ordinateur écoute sur un port. Cependant, votre box n'en a pas conscience et tous les messages qui vous seront adressés seront bloqués par la box qui agit comme un firewall et vous protège ainsi des intrusions. La redirection de port revient à dire à votre box que si une communication arrive sur le port XX (5000 dans le cas du projet), elle doit relayer le message à votre ordinateur.

7 Évaluation

30. Could you please summarize the conditions of the SID project ? When is the deadline ? *I have updated the subject of the project according to theses changes. Please refer to section 3 (p. 22)*

31. Vous nous aviez indiqué lors de la dernière séance qu'il y aurait des questions de compréhension du cours et de ce qui avait été fait en TP/TD. Y'aura-t-il des petits bouts de code à coder et/ou commenter ? Les questions seront-elles sous forme de QCM et/ou réponse libre ? *Il s'agira de questions courtes et indépendantes. Elles pourront concerner le cours ou ce qu'on a fait en TD/projet. La plupart seront sous forme de QCM ou corrigeables automatiquement. Mais il est possible qu'il y ait des questions qui demandent de rédiger 2 à 5 lignes de texte ou de code ou un bout de code à commenter/expliciter. (Par exemple, le code d'une route d'un serveur Flask en vous demandant quelle est l'url à interroger pour y accéder, interroger une base de données, etc.)*

32. Auriez-vous des précisions quant aux modalités de cet examen ? Faudra-t-il se connecter sur Zoom/Discord pendant l'examen ou simplement effectuer le test sur Moodle à l'horaire indiqué ? *Le test devrait se dérouler sur moodle, à l'heure indiquée avec contrainte d'être connecté sur zoom*[2]*, si possible avec une caméra différente de celle de votre ordi. Quoi qu'il en soit, je vous enverrai dans le week-end (probablement samedi soir), une explication des règles matérielles de déroulement de l'examen.*

2. ne serait-ce que pour pouvoir m'indiquer en temps réel si vous avez un souci technique