IMPLEMENTAÇÃO DE UMA BASE DE DADOS PARA FICHEIROS GENBANK

JOANA ARAÚJO PG49836, MARIANA SILVA PG45966, TIAGO SILVA PG49849

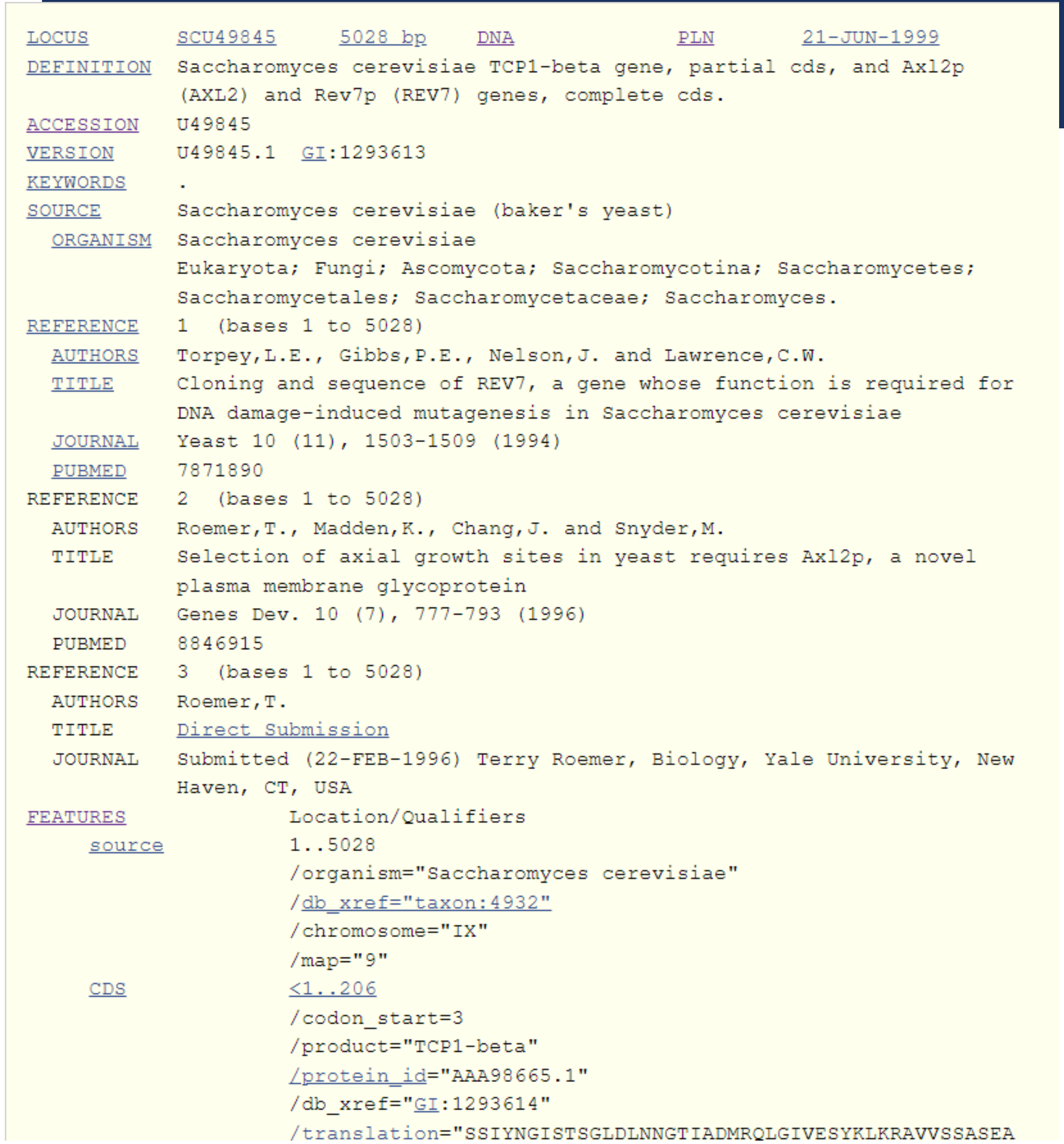
CONTEXTUALIZAÇÃO

Big data

- Extremamente úteis no mundo moderno onde somos constantemente bombardeados com informações;

Base dados-MySQL

- Modelo de dados mais flexível, maior escalabilidade;
- Desempenho superior, características de base dados relacionais.



```
LOCUS       SCU49845     5028 bp    DNA             PLN       21-JUN-1999
DEFINITION  Saccharomyces cerevisiae TCP1-beta gene, partial cds, and Axl2p
            (AXL2) and Rev7p (REV7) genes, complete cds.
ACCESSION   U49845
VERSION     U49845.1  GI:1293613
KEYWORDS    .
SOURCE      Saccharomyces cerevisiae (baker's yeast)
  ORGANISM  Saccharomyces cerevisiae
            Eukaryota; Fungi; Ascomycota; Saccharomycotina; Saccharomycetes;
            Saccharomycetales; Saccharomycetaceae; Saccharomyces.
REFERENCE   1  (bases 1 to 5028)
  AUTHORS   Torpey,L.E., Gibbs,P.E., Nelson,J. and Lawrence,C.W.
  TITLE     Cloning and sequence of REV7, a gene whose function is required for
            DNA damage-induced mutagenesis in Saccharomyces cerevisiae
  JOURNAL   Yeast 10 (11), 1503-1509 (1994)
  PUBMED    7871890
REFERENCE   2  (bases 1 to 5028)
  AUTHORS   Roemer,T., Madden,K., Chang,J. and Snyder,M.
  TITLE     Selection of axial growth sites in yeast requires Axl2p, a novel
            plasma membrane glycoprotein
  JOURNAL   Genes Dev. 10 (7), 777-793 (1996)
  PUBMED    8846915
REFERENCE   3  (bases 1 to 5028)
  AUTHORS   Roemer,T.
  TITLE     Direct Submission
  JOURNAL   Submitted (22-FEB-1996) Terry Roemer, Biology, Yale University, New
            Haven, CT, USA
FEATURES             Location/Qualifiers
     source          1..5028
                     /organism="Saccharomyces cerevisiae"
                     /db_xref="taxon:4932"
                     /chromosome="IX"
                     /map="9"
     CDS             <1..206
                     /codon_start=3
                     /product="TCP1-beta"
                     /protein_id="AAA98665.1"
                     /db_xref="GI:1293614"
                     /translation="SSIYNGISTSGLDLNNGTIADMRQLGIVESYKLKRAVVSSASEA
```

GENBANK

- Vasta biblioteca pública de sequências de proteínas e nucleótidos;
- É anotada com informação biológica e bibliográfica;
- Está está dividida em:

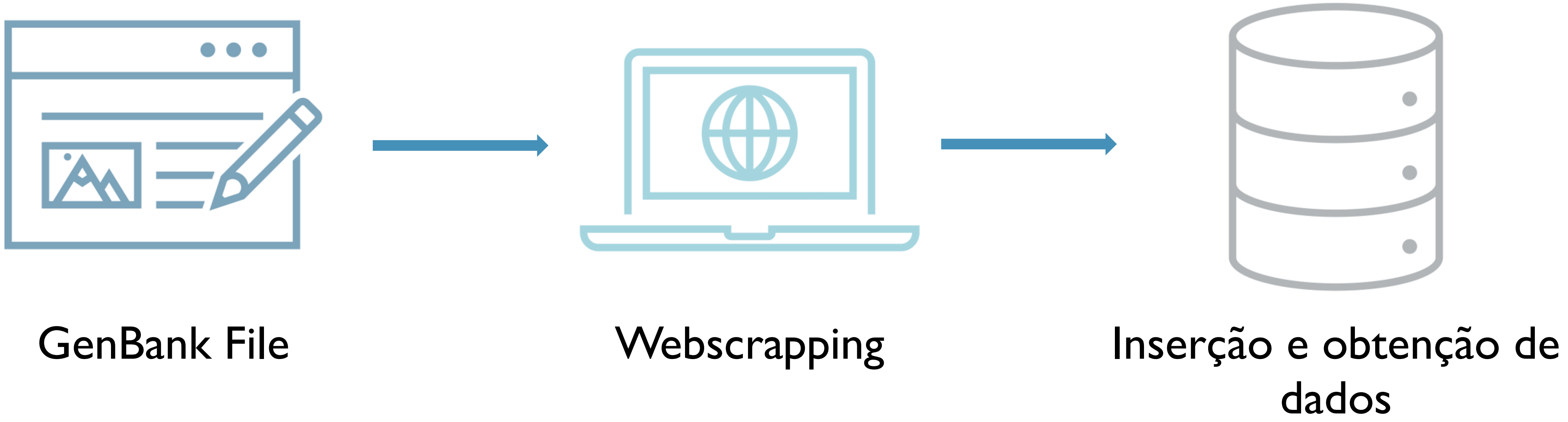
  -descrição da sequência;

  -nome científico;

  -taxonomia;

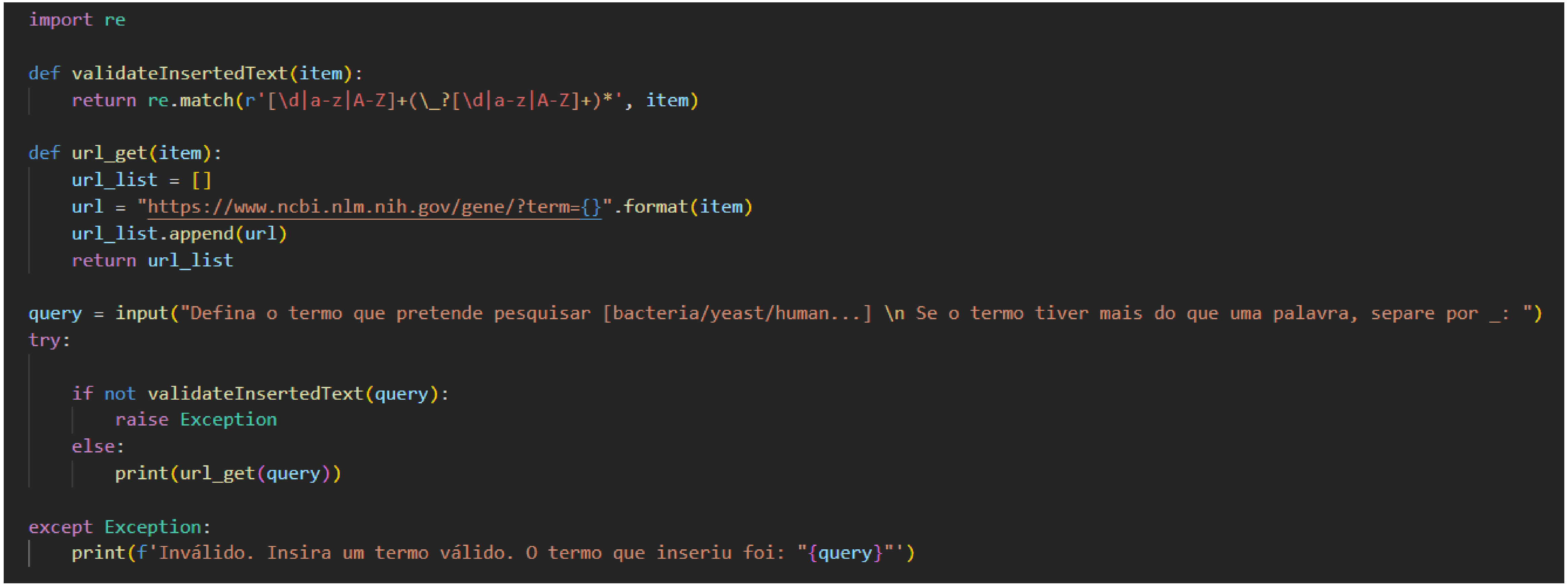
  -citações bibliográficas;

  -tabela de características.

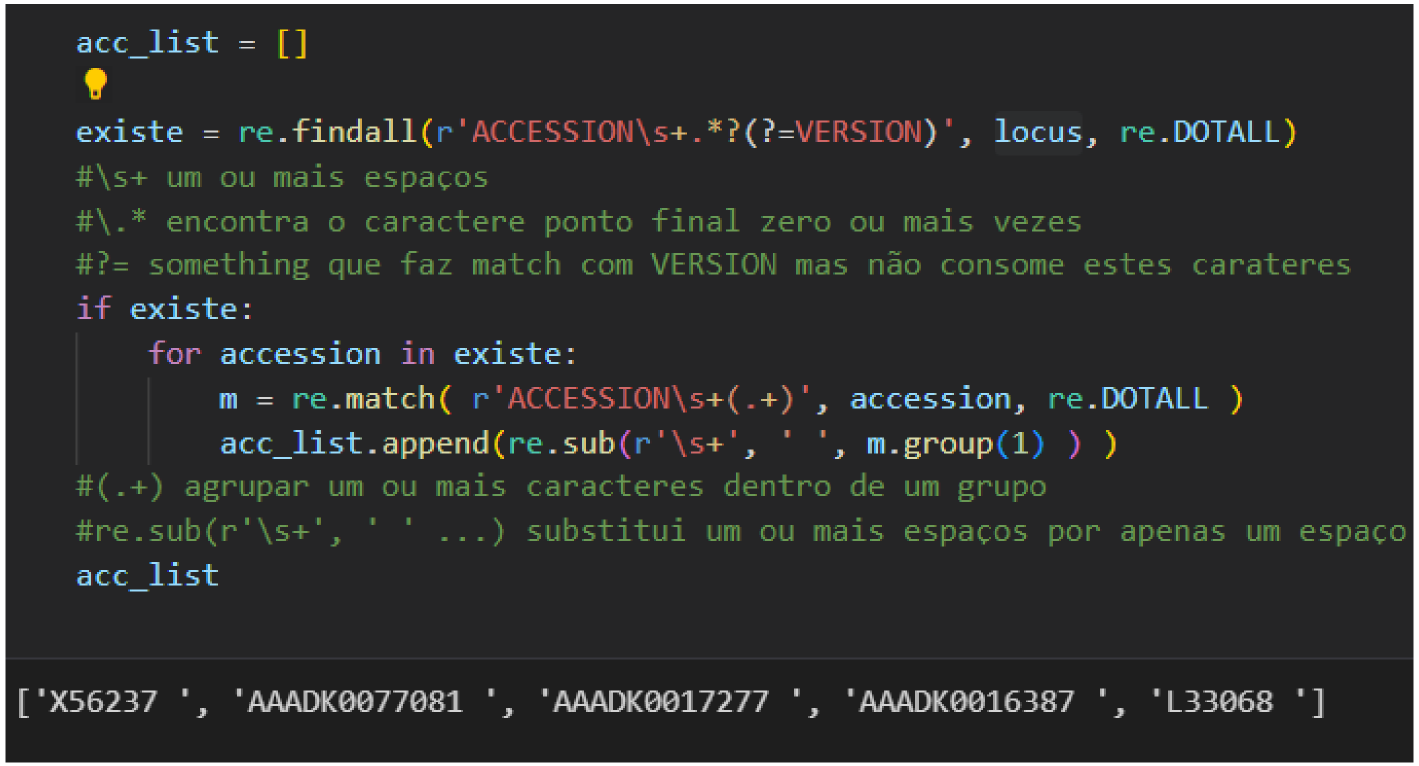


OBJETIVOS DO PROJETO

PERFIL DE UTILIZAÇÃO

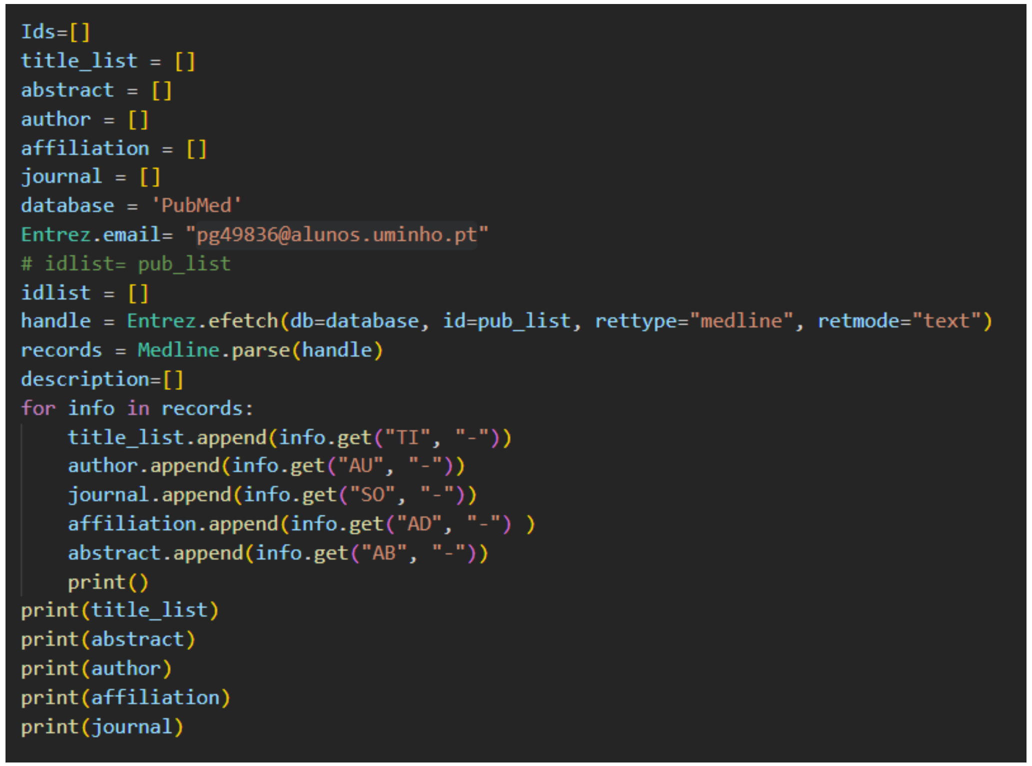


```python
import re

def validateInsertedText(item):
    return re.match(r'[\d|a-z|A-Z]+(\_?[\d|a-z|A-Z]+)*', item)

def url_get(item):
    url_list = []
    url = "https://www.ncbi.nlm.nih.gov/gene/?term={}".format(item)
    url_list.append(url)
    return url_list

query = input("Defina o termo que pretende pesquisar [bacteria/yeast/human...] \n Se o termo tiver mais do que uma palavra, separe por _: ")
try:

    if not validateInsertedText(query):
        raise Exception
    else:
        print(url_get(query))

except Exception:
    print(f'Inválido. Insira um termo válido. O termo que inseriu foi: "{query}"')
```



```
acc_list = []

existe = re.findall(r'ACCESSION\s+.*?(?=VERSION)', locus, re.DOTALL)
#\s+ um ou mais espaços
#\.* encontra o caractere ponto final zero ou mais vezes
#?= something que faz match com VERSION mas não consome estes carateres
if existe:
    for accession in existe:
        m = re.match( r'ACCESSION\s+(.+)', accession, re.DOTALL )
        acc_list.append(re.sub(r'\s+', ' ', m.group(1) ) )
#(.+) agrupar um ou mais caracteres dentro de um grupo
#re.sub(r'\s+', ' ' ...) substitui um ou mais espaços por apenas um espaço
acc_list
```

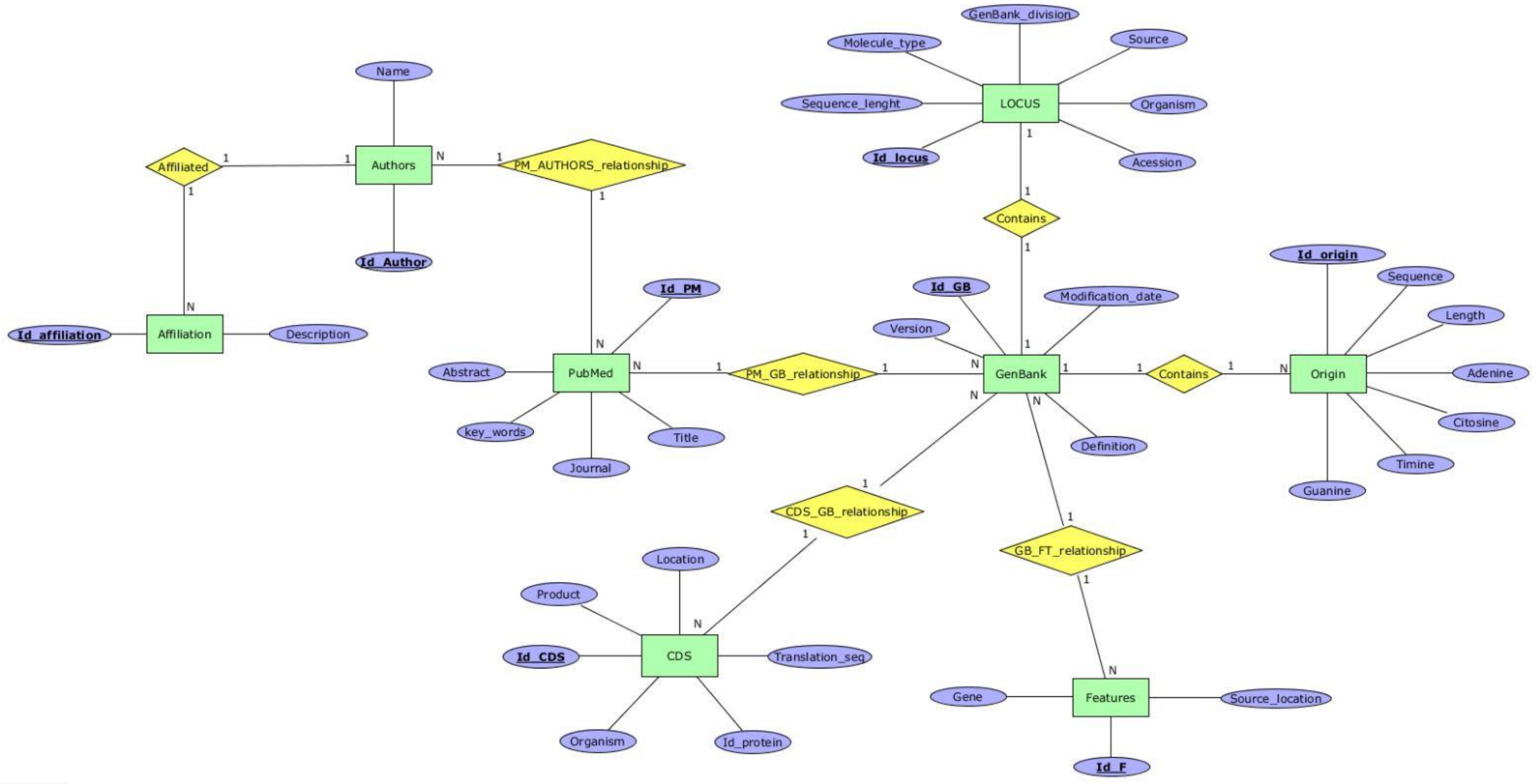
```
['X56237 ', 'AAADK0077081 ', 'AAADK0017277 ', 'AAADK0016387 ', 'L33068 ']
```



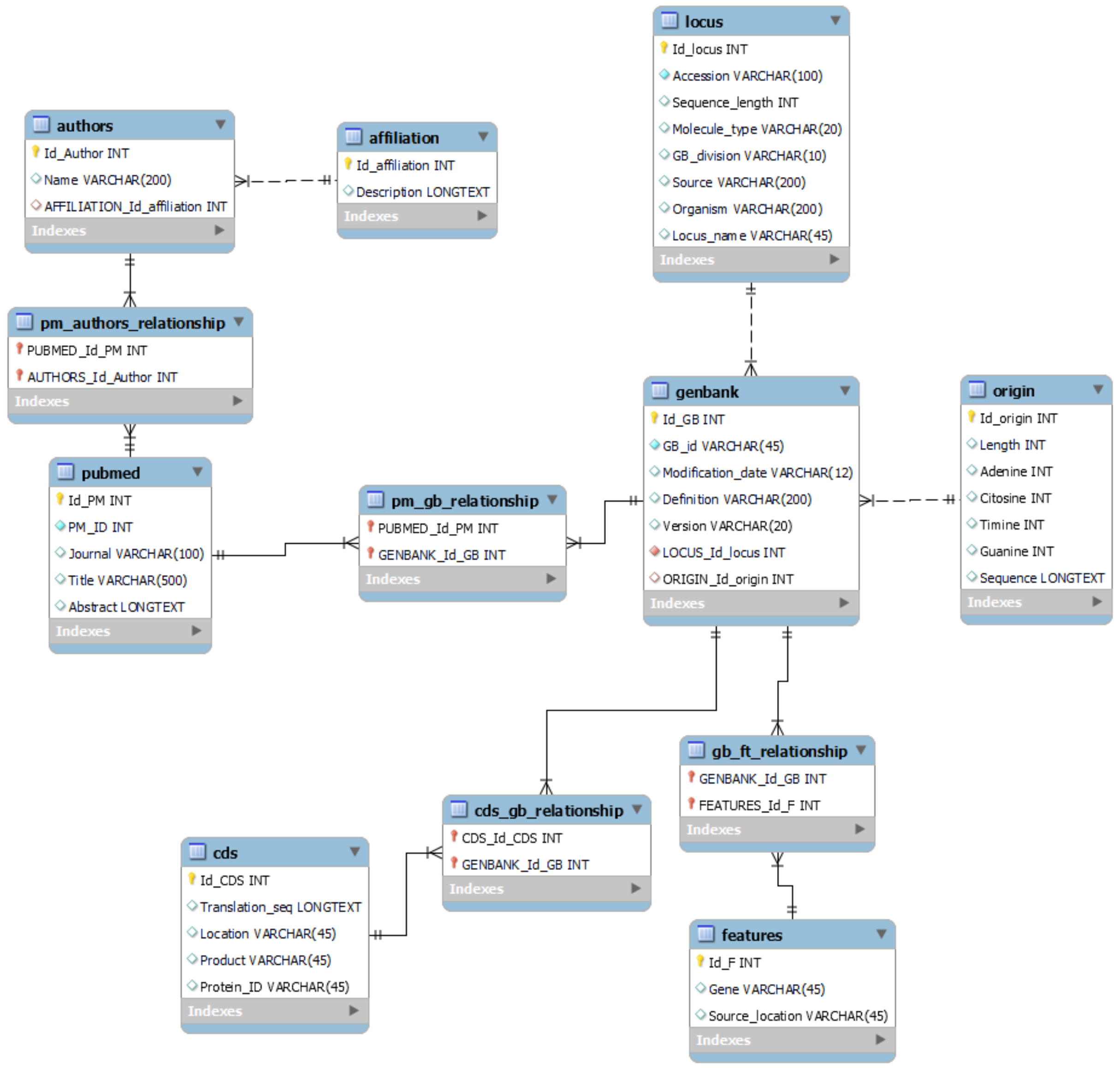
```
Ids=[]
title_list = []
abstract = []
author = []
affiliation = []
journal = []
database = 'PubMed'
Entrez.email= "pg49836@alunos.uminho.pt"
# idlist= pub_list
idlist = []
handle = Entrez.efetch(db=database, id=pub_list, rettype="medline", retmode="text")
records = Medline.parse(handle)
description=[]
for info in records:
    title_list.append(info.get("TI", "-"))
    author.append(info.get("AU", "-"))
    journal.append(info.get("SO", "-"))
    affiliation.append(info.get("AD", "-") )
    abstract.append(info.get("AB", "-"))
    print()
print(title_list)
print(abstract)
print(author)
print(affiliation)
print(journal)
```

EXPRESSÕES REGULARES E BIOPYTHON

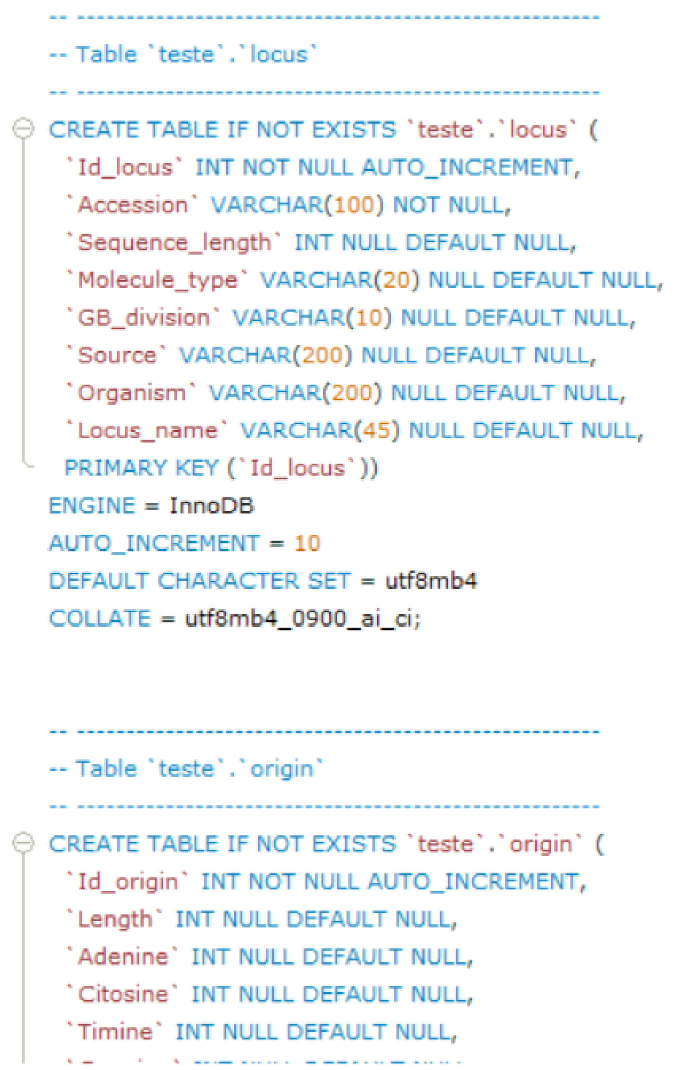
MODELO CONCEPTUAL



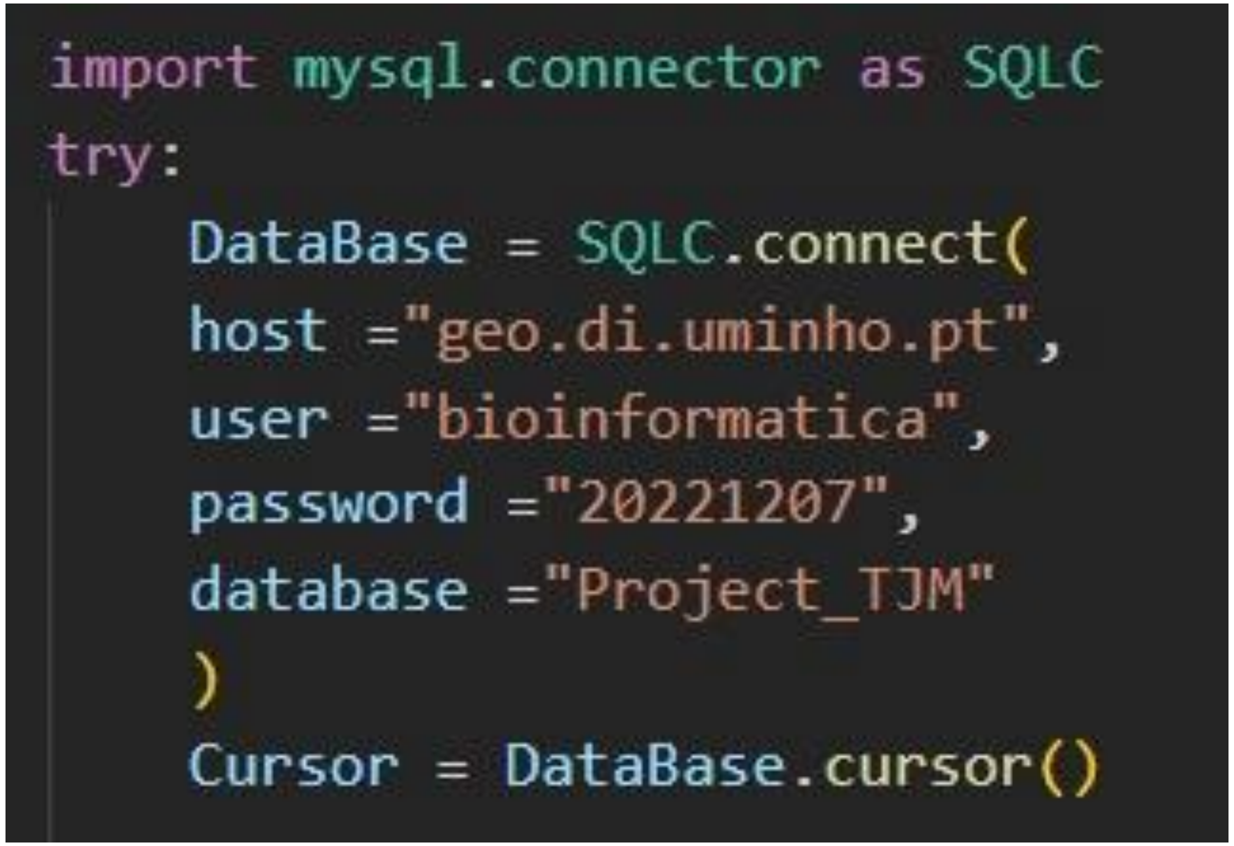
MODELO LÓGICO



MODELO FÍSICO

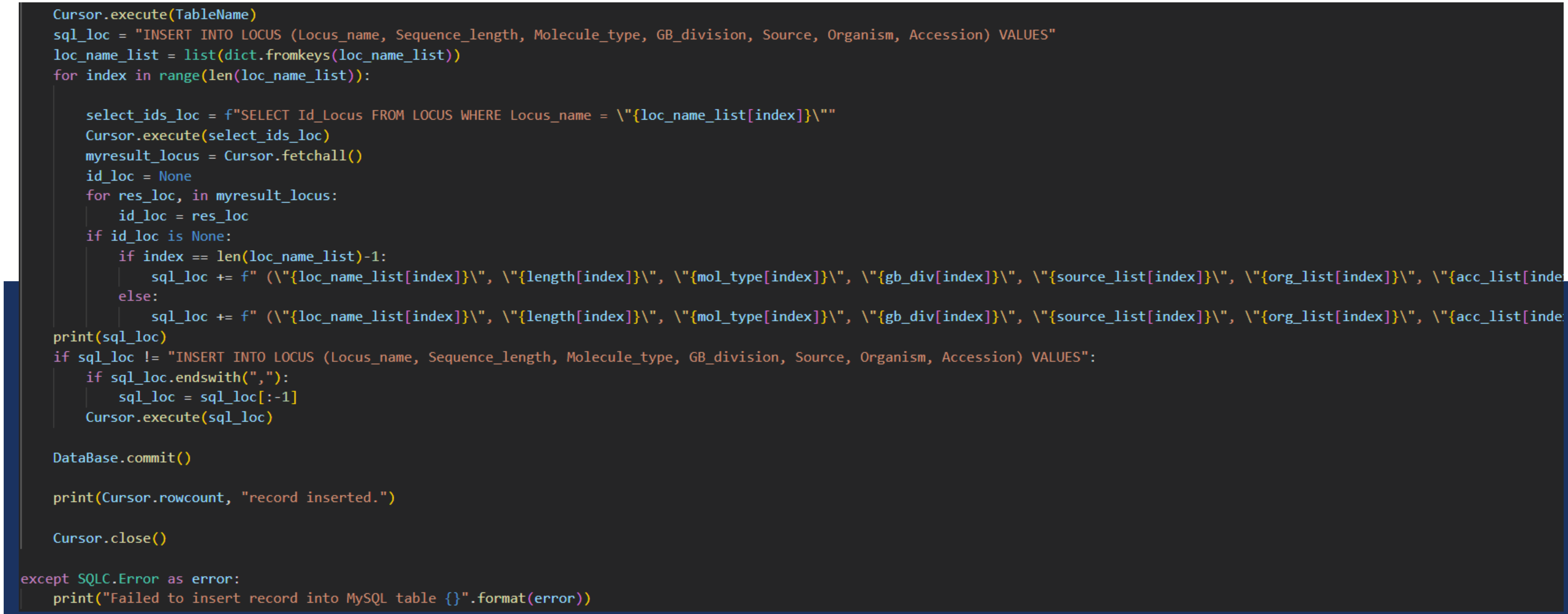


```
-- -----------------------------------------------------
-- Table `teste`.`locus`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `teste`.`locus` (
  `Id_locus` INT NOT NULL AUTO_INCREMENT,
  `Accession` VARCHAR(100) NOT NULL,
  `Sequence_length` INT NULL DEFAULT NULL,
  `Molecule_type` VARCHAR(20) NULL DEFAULT NULL,
  `GB_division` VARCHAR(10) NULL DEFAULT NULL,
  `Source` VARCHAR(200) NULL DEFAULT NULL,
  `Organism` VARCHAR(200) NULL DEFAULT NULL,
  `Locus_name` VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (`Id_locus`))
ENGINE = InnoDB
AUTO_INCREMENT = 10
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `teste`.`origin`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `teste`.`origin` (
  `Id_origin` INT NOT NULL AUTO_INCREMENT,
  `Length` INT NULL DEFAULT NULL,
  `Adenine` INT NULL DEFAULT NULL,
  `Citosine` INT NULL DEFAULT NULL,
  `Timine` INT NULL DEFAULT NULL,
```

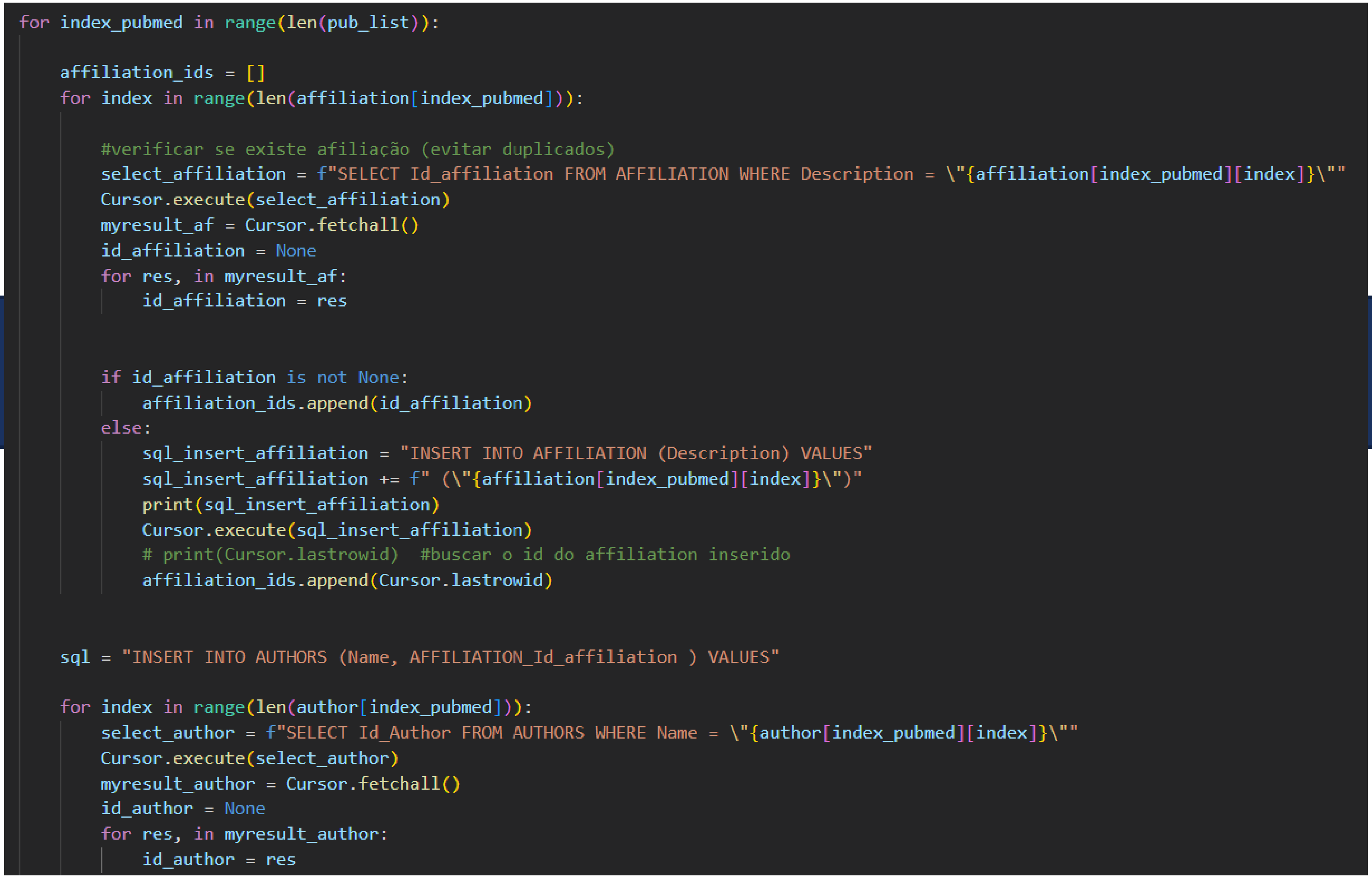


```
import mysql.connector as SQLC
try:
    DataBase = SQLC.connect(
    host ="geo.di.uminho.pt",
    user ="bioinformatica",
    password ="20221207",
    database ="Project_TJM"
    )
    Cursor = DataBase.cursor()
```

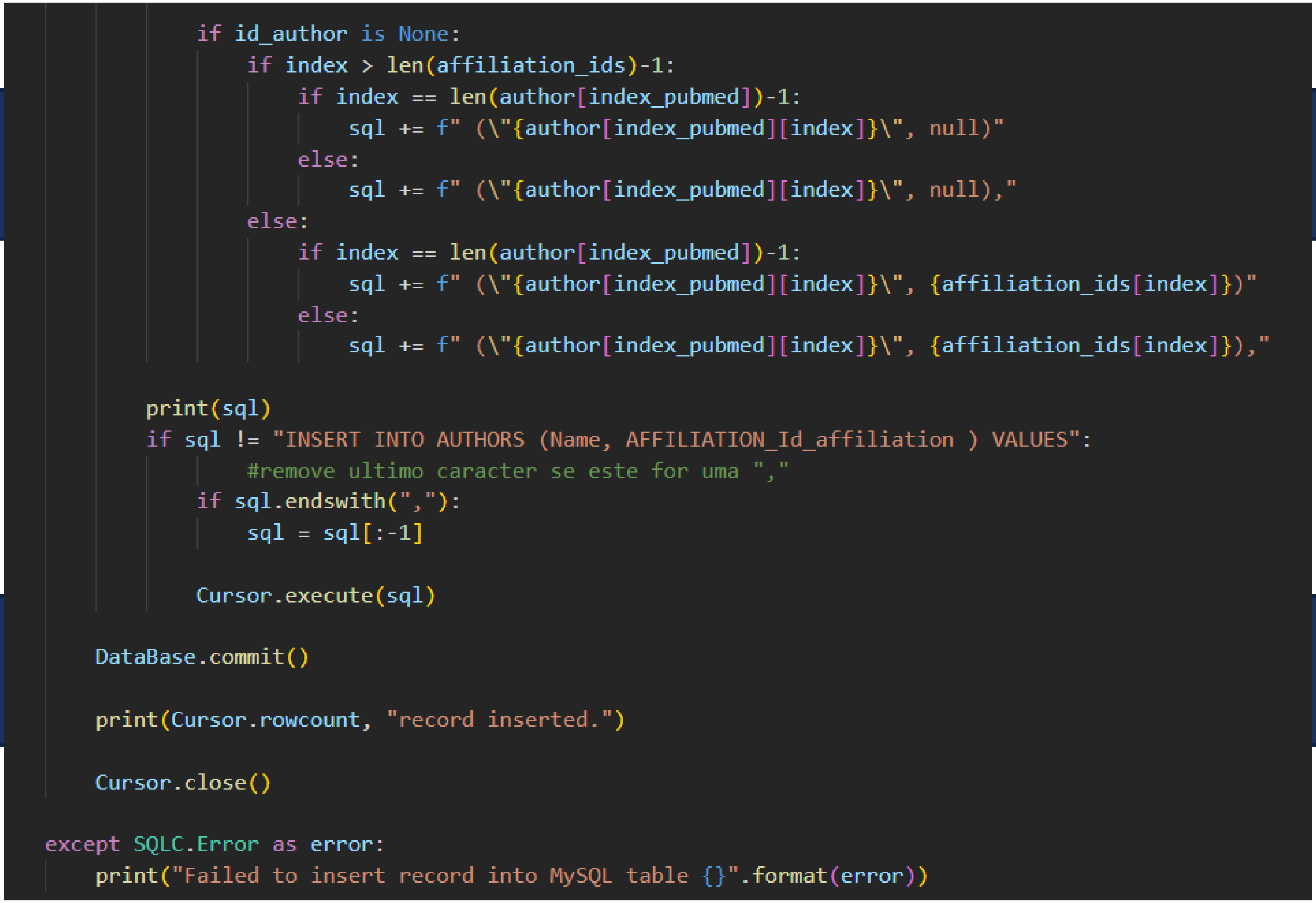
CONEXÃO À BASE DE DADOS



```
    Cursor.execute(TableName)
    sql_loc = "INSERT INTO LOCUS (Locus_name, Sequence_length, Molecule_type, GB_division, Source, Organism, Accession) VALUES"
    loc_name_list = list(dict.fromkeys(loc_name_list))
    for index in range(len(loc_name_list)):

        select_ids_loc = f"SELECT Id_Locus FROM LOCUS WHERE Locus_name = \"{loc_name_list[index]}\""
        Cursor.execute(select_ids_loc)
        myresult_locus = Cursor.fetchall()
        id_loc = None
        for res_loc, in myresult_locus:
            id_loc = res_loc
        if id_loc is None:
            if index == len(loc_name_list)-1:
                sql_loc += f" (\"{loc_name_list[index]}\", \"{length[index]}\", \"{mol_type[index]}\", \"{gb_div[index]}\", \"{source_list[index]}\", \"{org_list[index]}\", \"{acc_list[inde
            else:
                sql_loc += f" (\"{loc_name_list[index]}\", \"{length[index]}\", \"{mol_type[index]}\", \"{gb_div[index]}\", \"{source_list[index]}\", \"{org_list[index]}\", \"{acc_list[inde
    print(sql_loc)
    if sql_loc != "INSERT INTO LOCUS (Locus_name, Sequence_length, Molecule_type, GB_division, Source, Organism, Accession) VALUES":
        if sql_loc.endswith(","):
            sql_loc = sql_loc[:-1]
        Cursor.execute(sql_loc)

    DataBase.commit()

    print(Cursor.rowcount, "record inserted.")

    Cursor.close()

except SQLC.Error as error:
    print("Failed to insert record into MySQL table {}".format(error))
```



```python
for index_pubmed in range(len(pub_list)):

    affiliation_ids = []
    for index in range(len(affiliation[index_pubmed])):

        #verificar se existe afiliação (evitar duplicados)
        select_affiliation = f"SELECT Id_affiliation FROM AFFILIATION WHERE Description = \"{affiliation[index_pubmed][index]}\""
        Cursor.execute(select_affiliation)
        myresult_af = Cursor.fetchall()
        id_affiliation = None
        for res, in myresult_af:
            id_affiliation = res


        if id_affiliation is not None:
            affiliation_ids.append(id_affiliation)
        else:
            sql_insert_affiliation = "INSERT INTO AFFILIATION (Description) VALUES"
            sql_insert_affiliation += f" (\"{affiliation[index_pubmed][index]}\")"
            print(sql_insert_affiliation)
            Cursor.execute(sql_insert_affiliation)
            # print(Cursor.lastrowid)  #buscar o id do affiliation inserido
            affiliation_ids.append(Cursor.lastrowid)


    sql = "INSERT INTO AUTHORS (Name, AFFILIATION_Id_affiliation ) VALUES"

    for index in range(len(author[index_pubmed])):
        select_author = f"SELECT Id_Author FROM AUTHORS WHERE Name = \"{author[index_pubmed][index]}\""
        Cursor.execute(select_author)
        myresult_author = Cursor.fetchall()
        id_author = None
        for res, in myresult_author:
            id_author = res
```



```
            if id_author is None:
                if index > len(affiliation_ids)-1:
                    if index == len(author[index_pubmed])-1:
                        sql += f" (\"{author[index_pubmed][index]}\", null)"
                    else:
                        sql += f" (\"{author[index_pubmed][index]}\", null),"
                else:
                    if index == len(author[index_pubmed])-1:
                        sql += f" (\"{author[index_pubmed][index]}\", {affiliation_ids[index]})"
                    else:
                        sql += f" (\"{author[index_pubmed][index]}\", {affiliation_ids[index]}),"

        print(sql)
        if sql != "INSERT INTO AUTHORS (Name, AFFILIATION_Id_affiliation ) VALUES":
                #remove ultimo caracter se este for uma ","
            if sql.endswith(","):
                sql = sql[:-1]

            Cursor.execute(sql)

    DataBase.commit()

    print(Cursor.rowcount, "record inserted.")

    Cursor.close()

except SQLC.Error as error:
    print("Failed to insert record into MySQL table {}".format(error))
```

PERSPETIVAS FUTURAS

OBRIGADO!!!!