Pasos para concretar el proyecto

1. Conformar el grupo
2. Creación del repositorio
3. Crear la base de datos
4. Crear las tablas
5. Conectarse desde Python
6. Crear clases en Python y probar que funcionan
    a. Consultar datos (SELECT)
    b. Ingresar Datos (INSERT INTO)
    c. Modificar Datos (UPDATE)
    d. Eliminar datos (DELETE)
    e. Por último hacer commit con los cambios
7. Crear un menu que dispare los métodos de acuerdo a las opciones que elige el usuario
8. Escribir función Main que articula todas las partes

02 Creación del repositorio

- Grupos de entre 3 y 4 integrantes.
- Utilización y entrega mediante repositorio de GitHug.
- Trabajo colaborativo demostrado en el repositorio.
    - Una rama por integrante.
    - Sin roles, solo que hagan commits.
    - División de tareas y creación de issues.

Fecha de entrega: 04/11/2022

03 Crear la base de datos

```
CREATE DATABASE proyectoIntegradorv01;

USE proyectoIntegradorv01;
```

04 Crear las tablas

```
CREATE TABLE `Propietario` (
        `id_Propietario` INT,
        `Nombre` varchar(50),
        `Apellido` varchar(50),
        `FechaNacimiento` datetime,
    PRIMARY KEY (`id_Propietario`)
);
```

05 Conectarse desde Python

Acceso desde Python

En Python, el acceso a bases de datos se encuentra definido a modo de estándar en las especificaciones de DB-API, que puedes leer en la PEP 249. Esto, significa que independientemente de la base de datos que utilicemos, los métodos y procesos de conexión, lectura y escritura de datos, desde Python, siempre serán los mismos, más allá del conector.

En Python debemos instalar el conector para mysql

Si no tenemos pip instalarlo con:

```
$ curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
$ python get-pip.py

$ python -m pip install mysql-connector-python
```

Con este modulo instalado nos podemos conectar a la base de datos.

```
import mysql.connector

midb = mysql.connector.connect(
  host="localhost",
  user="usuario",
  password="contraseña",
  database="proyectointegradorv01"
)

print(midb)
```

Primero abrimos una conexión al servidor MySQL y almacenamos el objeto de conexión en la variable midb. Luego creamos un nuevo cursor, por defecto un objeto MySQLCursor, usando el método cursor() de la conexión.

```
cursor = midb.cursor()

cursor.execute("SHOW TABLES")

for x in cursor:
  print(x)
```

Por último cerramos la conexión

```
cursor.close()
midb.close()
```

Código final

```
import mysql.connector

def conectar():
    try:
        database = mysql.connector.connect(
            host = "localhost",
            port = 3306,
            user = "root",
            passwd = "root",
            database = "proyectoIntegradorv01"
            )
        cursor = database.cursor(buffered=True)
        return [database, cursor]

    except:
        print("Base de datos inacesible")
```

https://dev.mysql.com/doc/connector-python/en/connector-python-connecting.html

06 Crear clases en Python

```
class Tipo:

    Nombre_Tipo = ""

    def __init__(self, nombre_Tipo):
        self.Nombre_Tipo = nombre_Tipo

    def get_nombre_Tipo(self):
        return Nombre_Tipo

    def set_nombre_Tipo(self, nombre_Tipó):
        self.Nombre_Tipo = nombre_Tipo
```

```
...
```

Clase del profe Kevin
https://sc.tucampus.org/playback/presentation/2.3/a3e77a5f88f1fa9adf3a4b41866f9885fd297f13-1666306197092

https://docs.python.org/es/3/tutorial/classes.html

06a Consultar datos

```
query = ("SELECT first_name, last_name, hire_date FROM employees "
         "WHERE hire_date BETWEEN %s AND %s")

hire_start = datetime.date(1999, 1, 1)
hire_end = datetime.date(1999, 12, 31)

cursor.execute(query, (hire_start, hire_end))
```

https://dev.mysql.com/doc/connector-python/en/connector-python-example-cursor-select.html

06b Ingresar datos

```
add_employee = ("INSERT INTO employees "
                "(first_name, last_name, hire_date, gender, birth_date) "
                "VALUES (%s, %s, %s, %s, %s)")

add_salary = ("INSERT INTO salaries "
              "(emp_no, salary, from_date, to_date) "
              "VALUES (%(emp_no)s, %(salary)s, %(from_date)s, %(to_date)s)")

data_employee = ('Geert', 'Vanderkelen', tomorrow, 'M', date(1977, 6, 14))

# Insert new employee
cursor.execute(add_employee, data_employee)
```

https://dev.mysql.com/doc/connector-python/en/connector-python-example-cursor-transaction.html

06c Modificar datos

```
# Query to get employees who joined in a period defined by two dates
query = (
  "SELECT s.emp_no, salary, from_date, to_date FROM employees AS e "
  "LEFT JOIN salaries AS s USING (emp_no) "
  "WHERE to_date = DATE('9999-01-01')"
  "AND e.hire_date BETWEEN DATE(%s) AND DATE(%s)")

# UPDATE and INSERT statements for the old and new salary
update_old_salary = (
  "UPDATE salaries SET to_date = %s "
  "WHERE emp_no = %s AND from_date = %s")
insert_new_salary = (
  "INSERT INTO salaries (emp_no, from_date, to_date, salary) "
  "VALUES (%s, %s, %s, %s)")

# Select the employees getting a raise
curA.execute(query, (date(2000, 1, 1), date(2000, 12, 31)))
```

https://dev.mysql.com/doc/connector-python/en/connector-python-tutorial-cursorbuffered.html

07 Menu

De acuerdo a la opción que ingrese el usuario se ejecuta un metodo u otro de las clases creadas

-> Usar función print() para presentar las opciones
<- e input() para leer la opción elegida

luego estructura if-elif-else.

```
a = input("Ingrese la opción)
if a == 1:
    Codigo opcion 1
elif a == 2:
    Codigo opcion 2
elif a == 3:
    Codigo opcion 3
else:
    print ("La opcion elegida no es validad")
```

el código incompleto pero con el funcionamiento necesario para entenderlo es:

```
from os import system
import conexion
import propiedad

connect = conexion.conectar()
database = connect[0]
cursor = connect[1]

def menuppal():
  system("cls")
  print("**** Inmobiliaria Bienes Raices Future ****")
  print()

  opcion = input("""
                 1. Ingresar Propiedad
                 2. Modificar Propiedad
                 3. Eliminar Propiedad
                 4. Listados
                 Q. Salir
                 Seleccione una opcion: """)

  if opcion == "1":
    ingresarProp()
  elif opcion == "2":
    modificarProp()
  elif opcion == "3":
    listados()
  elif opcion == "Q" or opcion == "q":
    cursor.close()
    database.close()
    quit()
  else:
    print("Opcion no valida.")
    print("Por favor, intente nuevamente.")
    system("cls")
    return()

#Modificar para usar el método y atributos de la clase correspondiente
def ingresarProp():
  idProp = input("Ingresar ID ")
  nombre = input("Ingresar Propiedad ")
  direccion = input("Ingresar Direccion ")
  contacto = input("Ingresar Contacto ")

  agregarProp = ("INSERT INTO propiedad (id_Propiedad, Nombre, Direccion, Contacto) VALUES (%s, %s, %s, %s)")
  datosProp = (idProp, nombre, direccion, contacto)

  cursor.execute(agregarProp, datosProp)
  database.commit()
  print("Propiedad agregada")
  return()

def modificarProp():
  print("Modificar Propiedad")
  return()
```

08 Main

La función Main llama al menu y podríamos hacer otras tareas, como validar que se conecta a la base de datos, etc.

```
import menu
```

```
def main():
   menu.menuppal()

while(True):
   main()
```