# **Programación Avanzada de Computadoras**

## **Laboratorio Calificado 1**

### **2020-2**

### **Luis A. Muñoz**

**Nombre: *Victor Alonso Nuñez Cabrera***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Objetivo:**

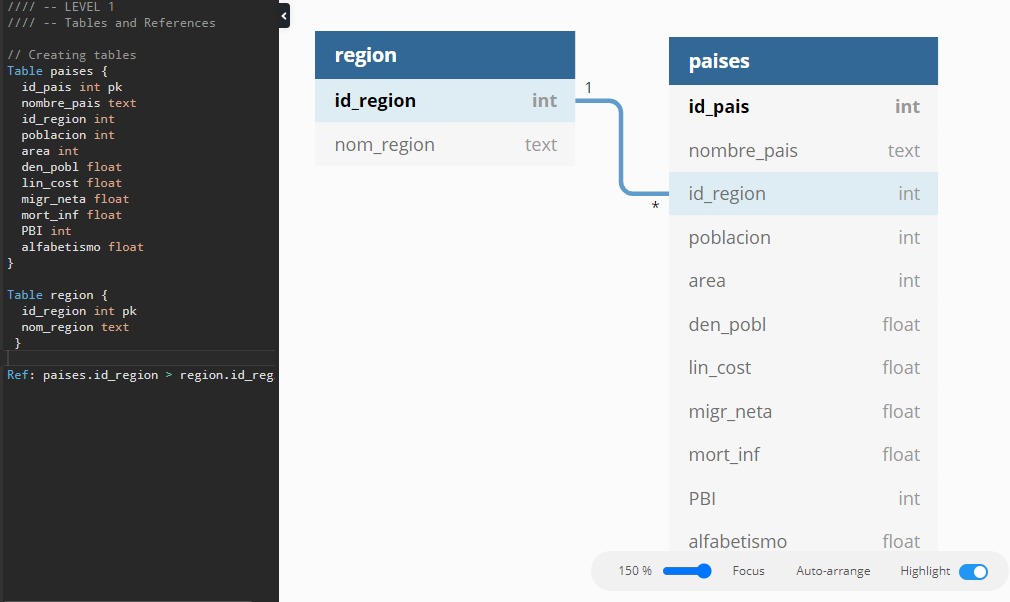
Diseñar e implementar una aplicación de reportes sobre estadísticas con datos mundiales a nivel de países y regiones.

**Diseño:**

Debe de construir una base de datos que contenga la información del archivo CSV con datos estadísticos del año 2019 de los países del mundo, *countries\_data.csv*. La información de este archivo tiene las siguientes columnas.

* Country - país
* Region - región
* Population - población
* Area (sq. mi.) - área
* Pop. Density (per sq. mi.) – densidad poblacional
* Coastline (coast/area ratio) – línea costera
* Net migration – migración neta
* Infant mortality (per 1000 births) – mortalidad infantil
* GDP ($ per capita) - PBI
* Literacy - alfabetismo

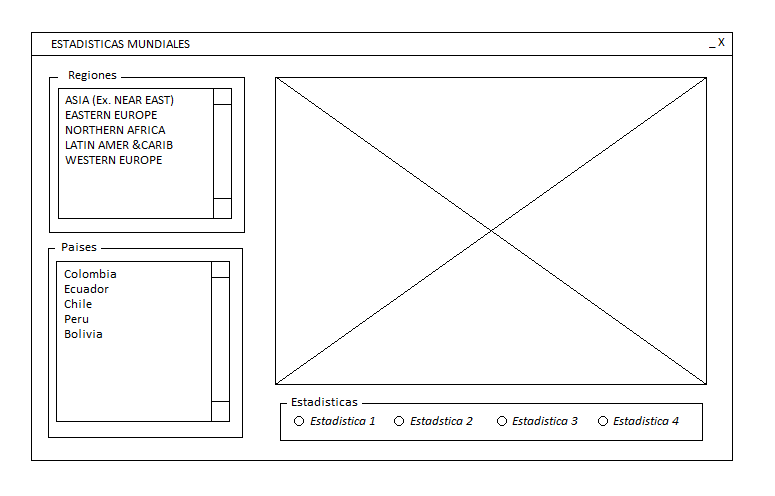
Organice esta información en dos tablas, una para regiones y otra para países. Especifique los campos, los tipos, así como las llaves primarias y foráneas que utilice su esquema. Diseñe su esquema utilizando la aplicación *dbdiagram.io* y adjúntelo como Figura 1.



*Figura 1 – Esquema de la base de datos (dbdiagram.io)*

Diseñe la clase **Database()** que implemente los métodos necesarios para el funcionamiento de su aplicación.

La aplicación tendrá una Interfaz Gráfica de Usuario (GUI) basada en *tkinter*. El diseño de la interfaz se muestra en el *wireframe* de la Figura 2.



*Figura 2 – Wireframe de la aplicación*

Esta interfaz tiene los siguientes elementos previamente definidos:

* *LabelFrame* con un *Listbox* de regiones, con todas las regiones de la base de datos
* *LabelFrame* con un *Listbox* de países, con todos los países de una región seleccionada en el *Listbox* de regiones.
* Ambos *Listbox* tienen un *Scrollbar* lateral para desplazamiento vertical
* *FigureCanvasTkAgg* para el soporte de las estadísticas por región.
* *LabelFrame* que contiene dos *Radiobutton* para cuatro estadísticas a personalizar.

Debe de reemplazar las etiquetas de los *Radiobuttons* por los nombres de los reportes que desea mostrar.

**Operación de la aplicación:**

Al iniciar la aplicación, el *FigureCanvasTkAgg* debe de estar en blanco (color de fondo de la figura: #F0F0F0) y ninguno de los *Radiobuttons* y de las opciones del *Listbox* de regiones debe de estar seleccionado. El *Listbox* de países debe de estar en blanco.

Cuando se seleccione una de las regiones, el *Listbox* de países debe de llenarse con los países que corresponda con la región seleccionada.

Al seleccionar unos de los *Radiobuttons*, se mostrará una de las cuatro estadísticas en caso de haber seleccionado una región (puede deshabilitar los *Radiobuttons* inicialmente y habilitarlos al seleccionar una región) en el *FigureCanvasTkAgg* que involucre todos los países de la región (por ejemplo, distribución poblacional por región).

Debe de diseñar cuatro estadísticas que pueda extraer de la base de datos en formato gráfico (gráficos de líneas, marcadores de datos, gráficos de barras, gráficos de pie o tarta, etc.). Detalle sus gráficos con leyendas para etiquetar los países por región.

**Entregables:**

Se deberán subir los siguientes archivos al Aula Virtual:

1. El presente documento con los datos completos (nombre completo, esquema y apéndice), convertido a archivo PDF.
2. Archivo *database.db* con la base de datos de su aplicación.
3. Archivo script *data\_app.py* con la solución (**no** un Jupyter Notebook).

Su archivo *script\_app.py* debe de contener la definición de la clase *Database*, así como la clase que contendrá la definición de la aplicación en *tkinter*. Considere que el archivo *database.db* estará en la misma ruta que el archivo de la aplicación. **No** incluya los pasos realizados para la creación de su archivo *database.db.* Esto lo documentará en el apéndice de este documento.

**Fecha máxima de entrega:**

Viernes 25 de setiembre de 2020, vía Aula Virtual (*Evaluaciones > Laboratorio Calificado 1*)

**Rúbrica de calificación:**

|  |  |  |  |
| --- | --- | --- | --- |
| Criterio | Nivel 1 | Nivel 2 | Nivel 3 |
| Diseña un esquema con las tablas que incluye campos, tipos y relaciones entre los campos, documentado en la *Figura 1* | No utiliza un base de datos para su solución **(0)** | Distribuye sus datos en una tabla con todos los registros **(1)** | Distribuye sus datos en diferentes tablas y las organiza de forma jerárquica **(2)** |
| Escribe las instrucciones SQL que construyen el esquema de las tablas diseñadas (Apéndice) | No construye una base da datos para la aplicación. **(0)** | Define instrucciones que construyen su tabla en sqlite3 de forma sencilla **(1)** | Define instrucciones que construyen su tabla en sqlite3 con las restricciones necesarias y las llaves y referencias correctas **(2)** |
| Carga los datos a partir del archivo CVS en las tablas | No sube sus datos a una base de datos **(0)** | Carga la información del archivo por un proceso manual **(1)** | Carga la información del archivo extrayendo de forma algorítmica los datos del archivo CVS **(2)** |
| Diseña el GUI utilizando *tkinter* según el diseño de la Figura 2 | No implementa todos los elementos del GUI **(0)** | Implementa los elementos del GUI, pero no están organizados según el *wireframe* **(3)** | Implementa los elementos del GUI según el *wireframe* y con las especificaciones iniciales **(4)** |
| Define y codifica la clase *Database* con los métodos requeridos para la aplicación. | No define una clase para el control de la base de datos **(0)** | Define una clase para el control de la base de datos, pero obvia un conjunto de opciones en el tratamiento de los datos **(2)** | Define una clase para el control de la base de datos, considerando criterios de ordenamiento y adecuación de resultados a retornar (listas, tuplas). **(3)** |
| La aplicación muestra la información de la base de datos (Regiones y Países) | La aplicación no muestra la lista de regiones y ni la lista de países **(0)** | La aplicación muestra la lista de regiones y la lista de países al seleccionar una región **(2)** | La aplicación muestra la lista de regiones y la lista de países al seleccionar una región, ambos datos ordenados de forma alfabética. **(3)** |
| La aplicación muestra los reportes de las diferentes regiones al seleccionar los *Radiobuttons*. | La aplicación no muestra reportes gráficos **(0)** | La aplicación muestra algunos reportes básicos **(2)** | La aplicación muestra reportes personalizados con todos los detalles (títulos, ejes, leyenda) **(4)** |

**Recursos:**

* Editor de esquema en línea: <https://dbdiagram.io/>
* Visor de archivos de base de datos: <https://sqliteonline.com/>
* Documentación *tkinter* en español: <https://guia-tkinter.readthedocs.io/es/develop/index.html>
* SQLite3: <https://likegeeks.com/es/tutorial-de-python-sqlite3/>

**APENDICE**

Documente las instrucciones que utilizó para construir los registros en la base de datos y la carga de la información a partir del archivo CSV (utilice un tipo de letra Consolas, tamaño 10).

CODIGO:

import os

import csv

import sqlite3

import tkinter as tk

import tkinter.ttk as ttk

import numpy as np

import matplotlib.pyplot as plt

ruta\_home = os.getcwd()

print(ruta\_home)

data = []

PATH = '.\CSV Files'

for item in os.listdir(PATH):

if item.endswith(".csv"):

#Lectura del archivo CSV

with open(PATH + '\\' + item, encoding = 'utf-8') as csv\_file:

reader = csv.reader(csv\_file, delimiter=",")

next(reader)

for idx,col in enumerate(reader):

if col[1].strip() == 'SUB-SAHARAN AFRICA': #N1

data\_col = ((idx,col[0].strip(),0,col[2].strip(),

col[3].strip(),col[4].strip(),col[5].strip(),

col[6].strip(),col[7].strip(),col[8].strip(),

col[9].strip()))

elif col[1].strip() == 'LATIN AMER. & CARIB':

data\_col = ((idx,col[0].strip(),1,col[2].strip(),

col[3].strip(),col[4].strip(),col[5].strip(),

col[6].strip(),col[7].strip(),col[8].strip(),

col[9].strip()))

elif col[1].strip() == 'ASIA (EX. NEAR EAST)':

data\_col = ((idx,col[0].strip(),2,col[2].strip(),

col[3].strip(),col[4].strip(),col[5].strip(),

col[6].strip(),col[7].strip(),col[8].strip(),

col[9].strip()))

elif col[1].strip() == 'NEAR EAST':

data\_col = ((idx,col[0].strip(),3,col[2].strip(),

col[3].strip(),col[4].strip(),col[5].strip(),

col[6].strip(),col[7].strip(),col[8].strip(),

col[9].strip()))

elif col[1].strip() == 'OCEANIA':

data\_col = ((idx,col[0].strip(),4,col[2].strip(),

col[3].strip(),col[4].strip(),col[5].strip(),

col[6].strip(),col[7].strip(),col[8].strip(),

col[9].strip()))

elif col[1].strip() == 'NORTHERN AMERICA':

data\_col = ((idx,col[0].strip(),5,col[2].strip(),

col[3].strip(),col[4].strip(),col[5].strip(),

col[6].strip(),col[7].strip(),col[8].strip(),

col[9].strip()))

elif col[1].strip() == 'C.W. OF IND. STATES':

data\_col = ((idx,col[0].strip(),6,col[2].strip(),

col[3].strip(),col[4].strip(),col[5].strip(),

col[6].strip(),col[7].strip(),col[8].strip(),

col[9].strip()))

elif col[1].strip() == 'WESTERN EUROPE':

data\_col = ((idx,col[0].strip(),7,col[2].strip(),

col[3].strip(),col[4].strip(),col[5].strip(),

col[6].strip(),col[7].strip(),col[8].strip(),

col[9].strip()))

elif col[1].strip() == 'NORTHERN AFRICA':

data\_col = ((idx,col[0].strip(),8,col[2].strip(),

col[3].strip(),col[4].strip(),col[5].strip(),

col[6].strip(),col[7].strip(),col[8].strip(),

col[9].strip()))

elif col[1].strip() == 'EASTERN EUROPE':

data\_col = ((idx,col[0].strip(),9,col[2].strip(),

col[3].strip(),col[4].strip(),col[5].strip(),

col[6].strip(),col[7].strip(),col[8].strip(),

col[9].strip()))

elif col[1].strip() == 'BALTICS':

data\_col = ((idx,col[0].strip(),10,col[2].strip(),

col[3].strip(),col[4].strip(),col[5].strip(),

col[6].strip(),col[7].strip(),col[8].strip(),

col[9].strip()))

data.append(data\_col)

#DATA FINAL DE PAISES = data

data\_regiones = []

data\_reg= []

PATH = '.\CSV Files'

for item in os.listdir(PATH):

if item.endswith(".csv"):

#Lectura del archivo CSV

with open(PATH + '\\' + item, encoding = 'utf-8') as csv\_file:

reader = csv.reader(csv\_file, delimiter=",")

next(reader)

for col in reader:

data\_regiones.append(col[1].strip())

data\_regiones = list(set(data\_regiones))

for idx, valor in enumerate(data\_regiones):

datos = ((idx,valor))

data\_reg.append(datos)

#DATA FINAL DE REGIONES = data\_reg

conn = sqlite3.connect('database.db')

cur = conn.cursor()

#Creacion de la tabla 'paises'

try:

cur.execute("DROP TABLE regiones")

conn.commit

except:

pass

sql = """CREATE TABLE IF NOT EXISTS regiones (id\_region INTEGER NOT NULL PRIMARY KEY,

nom\_region TEXT NOT NULL)"""

cur.execute(sql)

conn.commit()

#Se carga la información a partir de el archivo de los paises

sql = """INSERT INTO regiones(id\_region, nom\_region)

VALUES (?,?)"""

cur.executemany(sql,data\_reg)

conn.commit()

conn.close()

conn = sqlite3.connect('database.db')

cur = conn.cursor()

#Creacion de la tabla 'paises'

try:

cur.execute("DROP TABLE paises")

conn.commit

except:

pass

sql = """CREATE TABLE IF NOT EXISTS paises (id\_pais INTEGER NOT NULL PRIMARY KEY, nombre\_pais TEXT NOT NULL,\

id\_region INTEGER NOT NULL, poblacion INTEGER NOT NULL,\

area INTEGER NOT NULL, den\_pol FLOAT NOT NULL, lin\_cost FLOAT NOT NULL,\

migr\_neta FLOAT NOT NULL, mort\_inf FLOAT NOT NULL, pbi INTEGER NOT NULL,\

alfabetismo FLOAT NOT NULL,

FOREIGN KEY (id\_region) REFERENCES regiones(id\_region))"""

cur.execute(sql)

conn.commit()

#Se carga la información a partir de el archivo de los paises

sql = """INSERT INTO paises

(id\_pais, nombre\_pais, id\_region, poblacion, area,den\_pol,

lin\_cost, migr\_neta, mort\_inf,pbi,alfabetismo)

VALUES (?,?,?,?,?,?,?,?,?,?,?)"""

cur.executemany(sql,data)

conn.commit()

conn.close()