PROCESO GESTIÓN DEL TALENTO HUMANO

FORMATO INFORME MENSUAL EJECUCIÓN CONTRACTUAL

Barranquilla, Noviembre de 2021

Señor (a)
MANUEL GREGORIO HORMECHEA LANCE
SUPERVISOR(A) CONTRATO No. **2182213**
Coordinador Sede TIC
Automatización
Barranquilla

Asunto: Informe mensual de ejecución contractual
Mes Noviembre del año 2021

Referencia: No 2182213 del año 2021

LILIOLA PAOLA PRINS BALLESTAS, identificado con la cédula de ciudadanía No. 32.780.488 de Barranquilla, en mi calidad de Contratista del SENA, en Automatización, en cumplimiento del Contrato de Prestación de Servicios de la referencia, a continuación presento el Informe de actividades realizadas en el mes objeto de cobro.

Valor y forma de Pago: k). un pago correspondiente al mes de Noviembre de 2021 por valor de (3.376.526). Los honorarios serán pagados por el SENA al contratista de acuerdo al cronograma definido por la Dirección Administrativa y Financiera de la Dirección General, en la cuenta de Ahorros No. 620280107 BBVA, cuyo titular es el Contratista.

Plazo: Será hasta el 18de Diciembre de 2021.

OBJETO:
Prestar servicios profesionales como instructor para impartir formación profesional titulada y complementaria en el área de programación de software de acuerdo a la programación académica del SENA Centro Nacional Colombo Alemán.

Obligaciones Especificas:

No	Obligaciones	Acciones realizadas	Evidencias
1	Impartir formación a las fichas asignadas	Seleccionar las estrategias de enseñanza, aprendizaje y evaluación pertinentes para el grupo. Orientar el proceso de aprendizaje de acuerdo con el diseño curricular vigente. Reportar la información académica y administrativa que se generen en el proceso.	Se realizó solicitud de creación de las fichas.
2	Impartir formación a las fichas asignadas	Sesiones de orientación mediadas por tecnología (Google Meeting - WhatsApp)	Capturas de pantalla tomadas de las sesiones

3	Impartir formación a las fichas asignadas	Espacios Virtuales disponibles en la plataforma institucional	Materiales y proyectos publicados en la plataforma

A continuación relaciono los desplazamientos que realicé previo a la presentación de este informe. Una vez finalizado cada desplazamiento presenté al ordenador del gasto el informe en el Formato Informe Legalización Desplazamiento Contratista GTH-F-087, en el que se describieron las actividades desarrolladas y los resultados de cada desplazamiento. Cada informe cuenta con el visto bueno del Supervisor.
Se lista a continuación el soporte de la legalización de los desplazamientos realizados, los cuales forman parte integral del presente informe de ejecución contractual.

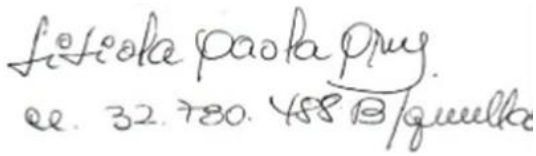
ITEM	No DE LA ORDEN DE VIAJE	LUGAR DE DESPLAZAMIENTO	FECHAS DE DESPLAZAMIENTOS	
			INICIO	FINALIZACIÓN

Nota 1: *Por cada desplazamiento que haya realizado el contratista, adjuntará el respectivo informe que la soporte. En caso de haber realizado el desplazamiento en fecha posterior a la presentación del informe de ejecución contractual, deberá reportarlo en el siguiente informe de ejecución contractual.*

Para el trámite de la cuenta me permito adjuntar: Documentos electrónicos enunciados como evidencias del cumplimiento de las obligaciones contractuales y los desplazamientos realizados y el No. 7814195051 de la planilla, operador y periodo. (Decreto Ley 2106 de 2019 – " *Decreto Ley Antitrámites"*)

Evidencias en () folios

Cordialmente,



LILIOLA PAOLA PRINS BALLESTAS
Contratista
C.C. No. 32.780.488

Recibí a satisfacción:

MANUEL GREGORIO HORMECHEA LANCE
Supervisor(a) Contrato 2182213 de 2021
Coordinador Sede TIC

ANEXOS – EVIDENCIAS.

CARACTERIZACIÓN DE RIESGO, CONTAGIO COVID-19

Como requisito para titularse del programa Técnico en Programación de Software, en el Centro Nacional Colombo Alemán (Servicio Nacional de Aprendizaje -SENA- Regional Atlántico), se hace necesario, que el aprendiz elabore un producto final que evidencie la aplicación de todas y cada una de las competencias desarrolladas durante su proceso formativo.

Para este grupo de formación, se trabajará con un proyecto que construya una herramienta en software (aplicación), de apoyo, al proceso de determinar el riesgo epidemiológico (caracterización de riego de contagio o infección), para un individuo, del SARSS-nCov2 (COVID-19).

Como primera fase del proyecto se debe determinar la relevancia (importancia) de cada uno de los siguientes factores como coadyuvante del riesgo de contagio, a nivel de un individuo o de un grupo de individuos.

Las variables para considerar son:

- Edad.
- Genero (sexo biológico).
- Ocupación.
- Dirección de residencia (Localidad).
- Morbilidades presentes.
- Raza
- Clima

Como ejemplo se ilustra la relación, en Colombia, de los contagios y la mortalidad en función del género (sexo biológico).



Ilustración 1. Contagios discriminados por género. Fuente INS.



Ilustración 2. Mortalidad discriminada por género. Fuente INS

La primera actividad del proyecto consiste en recolectar información publicada, por fuentes oficiales, a nivel local, regional, nacional e internacional, agrupar y combinar dicha información en función de las variables indicadas, de forma que permita construir el perfil de riesgo epidemiológico para un individuo.

A continuación, se suministran, a título de ejemplo, URLs de recursos en línea en los cuales se pueden encontrar estadísticas sobre la enfermedad por Coronavirus (COVID-19).

https://blogs.iadb.org/administracion-publica/es/estadisticas-oficiales-en-tiempos-del-coronavirus-continuidad-sin-contacto/

https://www.cepal.org/es/publicaciones/45962-disponibilidad-datos-estadisticos-enfrentar-la-pandemia-la-covid-19

https://www.dane.gov.co/index.php/estadisticas-por-tema/demografia-y-poblacion/informe-de-seguimiento-defunciones-por-covid-19

https://www.paho.org/bireme/index.php?option=com_content&view=article&id=471:pandemia-de-covid-19-as-estatisticas-de-acesso-a-bvs-e-o-alcance-da-cooperacao-tecnica-da-bireme&Itemid=183&lang=es

https://es.statista.com/temas/5901/el-coronavirus-de-wuhan/

P.S. se aconseja localizar algotras fuentes de información oficial, que contrasten y complementen la data suministrada y permitan una mejor caracterización de las variables objeto de estudio.

La pandemia por **COVID-19,** además de ser una emergencia sanitaria inédita para la humanidad, puso de manifiesto la capacidad de generar datos e información de forma masiva y en tiempo real, la mayoría de las naciones del mundo suministran diariamente, con diferentes niveles de detalles, información sobre contagios, ingresos hospitalarios, ingresos a unidades de cuidados intensivos (UCI) y defunciones.

La cantidad de información disponible permite ser analizada, a efectos de relacionar diferentes variables identificadas como coadyuvantes en el proceso de infección de un individuo. Dichos estudios pueden involucrar un metaanálisis cruzado de múltiples fuentes de información, con el fin de identificar las relaciones existentes entre diferentes variables, tales como la mortalidad en función de la edad, contagios en función del género, distribución geográfica de los contagios, entre otros. Dichos cruces posibilitan determinar el aporte de cada una de las variables identificadas a la posibilidad de contagio de un individuo; Adicionalmente se pueden utilizar técnicas y herramientas de la ciencia de datos, como la analítica predictiva para construir un modelo que estime la posibilidad (porcentaje) de riesgo de contagio.

Como parte del proyecto asignado, se propone realizar las siguientes actividades:

a) Con base en el análisis de las fuentes de datos suministradas la semana anterior, ¿Encontró dificultades a la hora de realizar agrupamiento de los datos, en función de alguna de las variables sugeridas? ¿Existe alguna otra variable que merezca ser incluida?
b) Inicie la construcción del documento metodológico, tomando como punto de referencia la siguiente tabla.

FASE	CONCEPTO
I	Conceptualización del problema de investigación
	- Definición y justificación del problema
	- Estado del conocimiento sobre el tema
	- Marco o modelo teórico
II	Elección del diseño
	- Hipótesis
	- Objetivos (General y Específicos)
III	Tipo de estudio
	- Observacional
	- Experimental

FASE	CONCEPTO
IV	Delimitación del estudio
	Población de estudio
	- Población objetivo
	- Población accesible
	- Criterios de Selección
	- Tipo de muestreo
	- Tamaño de la muestra
	Variables de estudio
	- Selección
	- Definición operativa
	- Definición funcional
	Recolección de datos
	- Fuentes de información
	- Método e instrumentos de medición
	- Calidad de los datos e instrumentos
	- Organización del trabajo de campo
	Estrategia de análisis
	- Proceso de los datos
	- Estrategias y técnicas de análisis
	Organización
	- Organización
	- Presupuesto
	- Aspectos éticos y legales
	Resultados
	- Presentación
	Conclusiones

Trabajaremos en una identificación preliminar de los siguientes aspectos: 1) Definición del problema, 2) Justificación del problema, 3) Objetivo General, 4) Objetivos específicos, 5) Población de estudio y 6) Recolección de datos.

c) Inicie un diseño a nivel grueso de los aspectos que se involucraran en el aplicativo, a saber: Seguridad (identificación de roles y usuarios) y Presentación de resultados.

Un joven aspiraba entrar a la marina y cuando acude a la entrevista de trabajo la primera pregunta que le hacen es ¿Sabe nadar?, a lo cual responde Y ¿para qué tienen los barcos?

LÓGICA DE PROGRAMACIÓN

Es la habilidad de pensar de manera razonada, sistemática y ordenada que nos hace capaces de inferir algoritmos abstractos y soluciones a problemas de negocios, susceptibles de programación.

Desde siempre se han utilizado ejercicios que favorecen el desarrollo de las habilidades lógicas, entre otros: las adivinanzas, sopas de letras, crucigramas y juegos de lógica. A continuación, se proponen algunos juegos de lógica con el fin de realizar el análisis del procedimiento que conduce a su solución.

EJERCICIO 1

Una joven asiste a una fiesta. Le presentan a cuatro hombres en rápida solución y, como es frecuente en tales situaciones, se menciona el trabajo que realiza cada uno de ellos. Desgraciadamente, a la joven le falla un poco la memoria y al cabo de una hora solo puede recordar que ha conocido a los señores Rubio, Castaño, Blanco y Moreno. Recuerda, además, que hay un fotógrafo, un tendero, un banquero y un cantante. Su anfitriona que es aficionada a los juegos de ingenio se niega a recordarle el nombre y la profesión de cada uno de los presentes , pero le proporciona las siguientes pistas:

1. El señor Blanco se encuentra interesado en que el banquero le realice un préstamo.
2. El señor Castaño conoció al fotógrafo cuando lo contrató para hacer las fotografía de su boda.
3. El cantante y el señor Blanco son amigos, pero nunca han tenido tratos de negocios.
4. Ni el señor Moreno ni el cantante conocían al señor Rubio antes de la fiesta.

Para resolver este ejercicio resulta importante Identificar el cruce de los datos suministrados, así como aplicar la lógica al momento de identificar afirmaciones y negaciones.

EJERCICIO 2

En un lago hay 7 piedras en línea y 6 ranas: 3 ranas macho en las 3 piedras de un lado, 3 ranas hembra en las 3 piedras del otro, y una piedra vacía central.

Las ranas pueden saltar a la piedra siguiente (si está vacía) o pueden saltar por encima de otra rana (del mismo sexo o contrario) y aterrizar 2 piedras más adelante (siempre que la piedra final esté vacía). No se pueden saltar dos ranas ni caer sobre una piedra ya ocupada. Las ranas pueden saltar hacia adelante y hacia atrás.

El reto consiste en conseguir que las ranas macho pasen a ocupar las piedras de las ranas hembra y viceversa.

URLs.
https://velneo.es/5-formas-mejorar-logica-programacion/
http://www.librosmaravillosos.com/los100mejoresjuegosdeingenio/index.html

```
# -*- coding: utf-8 -*-
import os
import random
class Persona():
    def __init__(self,Px,Py,St):
        self.X=Px
        self.Y=Py
        self.S=St

    def move(self,Dx,Dy,Max):
       Nx = self.X+Dx
       Ny = self.Y+Dy
       if(Nx<1):
           self.X = 1
       elif(Nx>Max):
           self.X = Max
       else:
           self.X = Nx

       if(Ny<1):
           self.Y = 1
       elif(Ny>Max):
           self.Y = Max
       else:
           self.Y = Ny

    def update(self,nSt):
        self.S=nSt

poblacion = []

def read_int(msg):
    try:
      return int(input(msg))
    except:
      return read_int('\n\tEl dato introducido no es correcto, vuelva a intentarlo
: ')

def ContarPoblacion(Px, Py):
    San = 0
    Inf = 0
    cpb = len(poblacion)
    for i in range(0,cpb):
        if(poblacion[i].X==Px and poblacion[i].Y==Py):
```

```
            if(poblacion[i].S=='I'):
                Inf = Inf+1
            else:
                San = San+1
    return [San,Inf]

def Infeccion(Px, Py):
    SwEstado = False
    i = 0
    cpb = len(poblacion)
    while (SwEstado==False and i<cpb):
        if(poblacion[i].X==Px and poblacion[i].Y==Py and poblacion[i].S=='I'):
            SwEstado = True
        else:
            i = i+1
    return SwEstado

def Infectar(Px, Py):
    can = 0
    cpb = len(poblacion)
    for i in range(0,cpb):
        if(poblacion[i].X==Px and poblacion[i].Y==Py and poblacion[i].S=='S'):
            poblacion[i].update('I')
            can = can+1
    return can

def Contar():
    san = 0
    inf = 0
    cpb = len(poblacion)
    for i in range(0,cpb):
        if(poblacion[i].S=='S'):
            san = san+1
        else:
            inf = inf +1
    return [san,inf]

def MostrarCiudad(Max):
    max = Max+1
    for i in range(1,max):
        print(' | ',end='')
        for j in range(1,max):
            [San,Inf] = ContarPoblacion(i,j)
            print(San,Inf,sep='::',end=' | ')
```

```
        print('\n')

def CalcularInfecciones(Max):
    cpb = len(poblacion)
    if cpb==0:
        print('\n\tNo existen individuos en la población')
    else:
        #
        # Se generan los movimientos al azar de la población
        #
        for i in range(0,cpb):
            Dx = random.randint(1,3)
            if(Dx==1):
                Dx = 1  # Intenta moverse hacia la derecha
            elif(Dx==2):
                Dx = 0  # Permanece en la columna donde se encuentra
            else:
                Dx = -1 # Intenta moverse hacia la izquierda
            Dy = random.randint(1,3)
            if(Dy==1):
                Dy = 1  # Intenta moverse hacia la abajo
            elif(Dy==2):
                Dy = 0  # Permanece en la fila donde se encuentra
            else:
                Dy = -1 # Intenta moverse hacia arriba
            poblacion[i].move(Dx, Dy, Max)
        #
        # Se calculan los infectados en función de las posiciones ocupadas
        #
        NIn = 0
        for i in range(0,cpb):
            Px = poblacion[i].X
            Py = poblacion[i].Y
            if Infeccion(Px,Py):
                NIn = NIn + Infectar(Px,Py)
        print('\n\tSe producen ',NIn,' nuevas infecciones')
    return True

def main():
    os.system("cls")
    TC = read_int('\n\tSeleccione el tamaño de la grilla (ciudad)
: ')
    PS = read_int('\n\tSeleccione la cantidad de personas sanas
: ')
```

```
    PI = read_int('\n\tSeleccione la cantidad de personas infectadas
: ')
    #
    # Se generan los individuos sanos y se ubican de forma aleatoria en la ciudad
    #
    for i in range(0,PS):
        PX = random.randint(1,TC)
        PY = random.randint(1,TC)
        NP = Persona(PX,PY,'S')
        poblacion.append(NP)
    #
    # Se generan los individuos infectados y se ubican de forma aleatoria en la ciu
dad
    #
    for i in range(0,PI):
        PX = random.randint(1,TC)
        PY = random.randint(1,TC)
        NP = Persona(PX,PY,"I")
        poblacion.append(NP)

    op = "N"
    os.system("cls")
    while op!="S" and op!="s":
        MostrarCiudad(TC)
        [San,Inf] = Contar()
        print('\n\tEn la ciudad quedan : ',San,' personas sanas y ',Inf,' personas
infectadas')
        if(San==0):
            print("\n\tLa infección se propagó al total de la población, ya no qued
an personas sanas")
        op = input("\n\tDesea finalizar? [S/N] : ")
        if(op=="n" or op=="N"):
            CalcularInfecciones(TC)
        elif(op=="S" or op=="s"):
            print("\n\tHasta la vista ...")
        else:
            print("\n\tSeleccione una opción válida ...")
    input("Enter para continuar")

main()
```

```
"""
La tienda del barrio, se encuentra intersada en sistematizar su
operación; para lo cual lo
contacta y le suministra los siguientes datos:

    a) Se permiten operaciones de venta y devolución
    b) Las ventas son anónimas
    c) Solo hay ventas a creditos
    d) No se puede vender mas de la cantidad en existencia de un
 producto
    e) Los productos se compran a los proveedores y se ingresan
al inventario
    f) En algunos casos se realizan devoluciones a los proveedor
es

PROBLEMA        :   Lentitud para determinar que hay en los inve
ntarios y que comprar

META            :   Sistematizar los procesos de la tienda

ANÁLISIS
    Entradas    :   a) Compras de los clientes

                    articulo, cantidad
                    Valor entregado para pagar

                    Valor de la cuenta    ==> Sistema
                    Valor del cambio (Vueltos) ==> Sistema

                    b) Compras a proveedores
                    articulo, cantidad
                    Factura de venta

                    c) Devolución de un cliente
```

```
                    articulo, cantidad

                    ==>
                    articulo, cantidad

                    ==> Nota contable

                    c) Devolución al proveedor
                    articulo, cantidad

                    ==>
                    articulo, cantidad, cortesías (posiblemente)

                    ==> Nota contable

    Salidas     :   a) Facturas para pago
                    b) Inventarios actualizados
                    b) Reportes que se requieran

    Procesos    :   venta a cliente
                    a) El cliente solicita el articulo y la cant
idad
                    b) La tienda verifica existencias del artícu
lo
                    c) Se entrega al artículo al cliente
                    d) Se descuentan las existencias del inventa
rio
                    f) Se acumula el valor en la facturación

DISEÑO.

    ARTICULOS. Contendra la información de los artículos.
```

```
    +-----------+--------------+---------+-----------+------
--+
    | Código    | Nombre       | Precio  | Existencia | Estad
o |
    +-----------+--------------+---------+-----------+------
--+
    |       100 | traperos     |   10000 |         10 |
A |
    +-----------+--------------+---------+-----------+------
--+
    |       101 | Cebolla      |    2000 |         25 |
A |
    +-----------+--------------+---------+-----------+------
--+

    100;traperos;10000;10;A

    ['100','traperos',10000,10,'A']


    VENTAS. Contendra la información de las ventas realizadas.

    ENCABEZADO. Contiene el resumen
    +--------+-----------+-------+--------+
    | Codigo | Fecha     | Valor | Estado |
    +--------+-----------+-------+--------+
    |    501 | 2021-06-12 | 20000 | A      |
    +--------+-----------+-------+--------+
    |    502 | 2021-06-12 | 12000 | A      |
    +--------+-----------+-------+--------+

    DETALLE. Contiene todos los registros de la venta
    +-------------+----------------+------------------+------
-----------+
```

```
    | Codigo_Venta | Código_Artículo | Cantidad_Articulo | Precio_Artículo |
    +--------------+-----------------+-------------------+-----------------+
    |          501 |             100 |                 2 |           10000 |
    +--------------+-----------------+-------------------+-----------------+
    |          502 |             100 |                 1 |           10000 |
    +--------------+-----------------+-------------------+-----------------+
    |          502 |             101 |                 1 |            2000 |
    +--------------+-----------------+-------------------+-----------------+

    MENU:
    1. Ingreso de artículos
    2. Compra de artículos (Proveedor)
    3. Venta de artículos (Clientes)
    4. Baja de un artículo
    5. Reporte de existencias
    X. Finalizar

    MACROALGORITMO.
    a) Desplegar el menu
    b) Determinar la acción
    c) Ejecutar la acción

    INGRESO DE ARTICULOS
    a) Leer los datos del artículo (Codigo, Nombre, Precio, Existencias)
    b) Verificar que el artículo no exista en el archivo
```

```
    c) Adicionar los datos al archivo

    NOTA: Se trabajará el archivo con formato CSV, y el separado
r será ;

"""
import os
import time
import sys
#
# Programa principal, despliega el menu y permite selecionar la
opción correspondiente
#

##
# def ingreso(). Se encarga de registrar un nuevo producto en la
 tienda
#
#   Parametro de entrada
#       None
#
#   Valor de retorno
#                   ::  texto, indicando el resultado de la oper
ación
#   Forma de operación
#
#       a)  Se captura el código del nuevo registro
#       b)  Se realiza una búsqueda de ese código en el archivo
de datos
#       c)  Si se encuentra se indicará que no se puede añadir e
l registro (Código duplicado)
#           Si no se encuentra
#               c1) Se toma el resto de los datos
#               c2) Se añade el registro al archivo
```

```
#    Este corresponde a la letra C (Create)
def ingreso() -> str:
    try:
        sal = 'ERROR: El registro no pudo ser añadido.'
        cod = input('Suministre el código para el artículo
 : ')
        # Si no existe el archivo se crea
        if not os.path.exists('articulos.dat'):
            open('articulos.dat','x')
        with open('articulos.dat','rt') as f:
            # Representan una operación Read
            sw_enc = False    # Se supone que no se encontró
            for reg in f:
                dat = reg.split(';')  # el Código queda en dat[0
]
                if cod==dat[0]:
                    sw_enc = True   # Lo encontré
                    break
        if sw_enc:
            sal = 'ERROR: No se puede añadir el registro (Código
 duplicado)'
        else:
            nom = input('Suministre el nombre para el artículo
     : ').capitalize()
            pre = float(input('Suministre el precio para el artí
culo       : '))
            exi = float(input('Suministre las existencias para e
l artículo : '))
            with open('articulos.dat','at') as f:
                f.write(f'{cod};{nom};{pre};{exi};A\n')
                sal = 'INFO : Registro añadido con éxito'
    except Exception:
        pass
    return sal
```



```
##
# compra() : Representa las compras que la tienda realiza a sus
proveedores
#
#   Parametros de entrada
#       None
#
#   Valor de retorno
#                   ::  Texto, indicando el resultado de la oper
ación
#
#   a)  Leer el código del artículo y la cantidad
#   b)  Verificar que la cantidad sea > 0
#   c)  Ubicar el registro en el archivo
#   d)  Si no se encuentra el registro indicar error
#       Si se encuentra
#           d1) Incrementar la cantidad en existencias del artíc
ulo
#   e) Solicitar si se desea procesar mas registros
#
#   NOTA :  Se implementara de forma rudimentaria, cargando todo
 el archivo a memoria
#           Y realizando el proceso sobre la memoria. Al final s
e reescribe el archivo
#
#   Representan la operación U (Update)
def compra() -> str:
    try:
        # Cargar todo el archivo en memoria
        art = []
        with open('articulos.dat','rt') as f:
            art = f.readlines()
        rpta = 'S'
```

```
        sal = 'INFO: Operación realizada satisfactoriamente'
        while rpta=='S':
            cod = input('Suministre el código para el artículo
    : ')
            # Estas líneas ubicar el articulo
            pos = -10    # Significa no ha sido ubicado
            # Representan una R
            idx = 0
            for ele in art:
                dat = ele.replace('\n','').split(';')
                if dat[0]==cod and dat[4]=='A':
                    pos = idx
                    break
                idx += 1
            if pos==-10:   # No lo encontró
                print("ERROR: Artículo no encontrado o se encuen
tra inactivo")
                sal = 'ERROR: Se presentaron errores en el proce
so, por favor verifique'
            else:
                dat = art[pos].replace('\n','').split(';')
                dat[2] = float(dat[2])
                dat[3] = float(dat[3])
                print(f'Artículo       : {dat[1]}')
                print(f'Precio         : {dat[2]:10.2f}')
                print(f'Existencias    : {dat[3]:10.2f}')
                print(f'Estado         : {dat[4]}')
                exi = float(input('Suministre la cantidad adquir
ida para el artículo : '))
                if exi<0:
                    print("ERROR: La cantidad no pueden ser nega
tiva")
                else:
                    dat[3] += exi
```

```
                    art[pos] = f'{dat[0]};{dat[1]};{dat[2]};{dat
[3]};{dat[4]}\n'
            rpta = input('Desea procesar mas registros : ').uppe
r()
        with open('articulos.dat','wt') as f:
            f.writelines(art)
    except Exception:
        pass
    return sal

##
# venta() : Representa las ventas que la tienda realiza a sus cl
ientes
#
#   Parametros de entrada
#       None
#
#   Valor de retorno
#                   ::  Texto, indicando el resultado de la oper
ación
#
#   a)  Leer el código del artículo y la cantidad
#   b)  Verificar que la cantidad sea > 0 y menor que las existe
ncias
#   c)  Ubicar el registro en el archivo
#   d)  Si no se encuentra el registro indicar error
#       Si se encuentra
#           d1) Decrementar la cantidad en existencias del artíc
ulo
#   e) Solicitar si se desea procesar mas registros
#
#   NOTA :  Se implementara de forma rudimentaria, cargando todo
 el archivo a memoria
```



```
#           Y realizando el proceso sobre la memoria. Al final s
e reescribe el archivoç
#   OTRA NOTA: se propone a los estudiantes calcular e imprimir
el valor a pagar.
#
def venta() -> str:
    try:
        # Cargar todo el archivo en memoria
        art = []
        with open('articulos.dat','rt') as f:
            art = f.readlines()
        rpta = 'S'
        sal = 'INFO: Operación realizada satisfactoriamente'
        apagar = 0
        while rpta=='S':
            cod = input('Suministre el código para el artículo
     : ')
            # Estas líneas ubicar el articulo
            pos = -10    # Significa no ha sido ubicado
            for idx,ele in enumerate(art):
                dat = ele.replace('\n','').split(';')
                if dat[0]==cod and dat[4]=='A':
                    pos = idx
                    break
            if pos==-10:   # No lo encontró
                print("ERROR: Artículo no encontrado o se encuen
tra inactivo")
                sal = 'ERROR: Se presentaron errores en el proce
so, por favor verifique'
            else:
                dat = art[pos].replace('\n','').split(';')
                dat[2] = float(dat[2])
                qty = dat[3] = float(dat[3])
                print(f'Artículo       : {dat[1]}')
```

```
                print(f'Precio          : {dat[2]:10.2f}')
                print(f'Existencias     : {qty:10.2f}')
                print(f'Estado          : {dat[4]}')
                exi = float(input('Suministre la cantidad adquir
ida para el artículo : '))
                if exi<0 or exi>qty:
                    print(f"ERROR: Cantidad invalida no puede se
r negativa y debe ser menor que {qty:10.2f}")
                else:
                    dat[3] -= exi
                    apagar += exi*dat[2]
                    art[pos] = f'{dat[0]};{dat[1]};{dat[2]};{dat
[3]};{dat[4]}\n'
            rpta = input('Desea procesar mas registros : ').uppe
r()
        with open('articulos.dat','wt') as f:
            f.writelines(art)
        if apagar>0:
            print(f"El valor a pagar es : {apagar:10.2f}")
    except Exception:
        pass
    return sal

##
#
#
#
#
#
# D ==> Borrado Lógico
def baja():
    pass

##
```

```
# Se diseñará un reporte que muestre las existencias en inventar
io, para ello se propone
# la siguiente estructura.  Se presentará ordenado por código
# Parámetro de entrada
#   ordenar     ::  Texto, indicará si se ordena por código o po
r nombre
#
#                           REPORTE DE EXISTENCIAS EN INVENTAR
IO
#
#  CÓDIGO   DESCRIPCIÓN                                CANTIDAD
VALOR   TOTAL
#
#
#
    SUMA TOTAL
def reporte(ordenar) -> None:
    try:
        # Como se requiere ordenar se cargan los datos en una ma
triz
        art = []
        with open('articulos.dat','rt') as f:
            for line in f:
                dato = line.replace('\n','').split(';')
                dato[2] = float(dato[2])
                dato[3] = float(dato[3])
                art.append(dato)
        # Se ordena el arreglo, si se desea ordenar por nombres
tocaría utilizar una función
        if ordenar=='S':
            llave = 1
        else:
            llave = 0
```

```
        art.sort(key=lambda x:x[llave]) # Se utilizan funciones
anonimas para indicar cual
                                        # columna se utilizará p
ara ordenar
        # Se imprimen los registros
        sumat = 0
        print('{:^100}'.format("REPORTE DE EXISTENCIAS EN INVENT
ARIO"))
        print('{:^10}  {:^50}  {:^10}  {:^10}  {:^12}'.format("C
ÓDIGO",'DESCRIPCIÓN', 'CANTIDAD','VALOR','TOTAL'))
        for ele in art:
            if ele[4]=='A':
                qty = ele[3]
                prc = ele[2]
                sumat += qty*prc
                print(f'{ele[0]:<10}  {ele[1]:<50}  {qty:10.2f}
 {prc:10.2f}  {qty*prc:12.2f}')
        tit = 'Total'
        lin = '-'*100
        print(f'{lin}\n{tit:<86}{sumat:14.2f}\n{lin}')
    except Exception:
        pass


try:
    rpta = "1"
    if not os.path.exists('articulos.dat'):
       open('articulos.dat','x')
    while rpta!="X":
        os.system('CLS')
        print("MENU PRINCIPAL")
        print("1. Ingreso de artículos (nuevos)")
        print("2. Compra de artículos (proveedor)")
        print("3. Venta de artículos (clientes)")
        print("4. Baja de artículos (eliminar)")
```

```
        print("5. Reportes")
        print("X. Finalizar")
        rpta = input("Seleccione la opción : ").upper()
        if rpta=="1":
            print(ingreso())
        elif rpta=="2":
            print(compra())
        elif rpta=="3":
            venta()
        elif rpta=="4":
            baja()
        elif rpta=="5":
            preg = input('Lo quieres ordenar por nombre del artí
culo : [S/N] : ').upper()
            reporte(preg)
        elif rpta!="X":
            print("Error: Opción no valida")
        time.sleep(3)
except Exception:
    pass
```

EJEMPLOS DE OTRAS APLICACIONES QUE INCLUYEN EL DESARROLLO DE VIDEO JUEGOS – CON Y SIN INTERFACES GRÁFICAS.

JUEGO DEL TRIQUI (TIC-TAC-TOE)

```
from tkinter import Tk, Button, messagebox
symbol = 'O'
board = [['','',''],['','',''],['','','']]

def gamestatus(status=False):
    global symbol
    global board
    if not status:
        btn00['state'] = 'disabled'
        btn01['state'] = 'disabled'
        btn02['state'] = 'disabled'
        btn10['state'] = 'disabled'
        btn11['state'] = 'disabled'
        btn12['state'] = 'disabled'
        btn20['state'] = 'disabled'
        btn21['state'] = 'disabled'
        btn22['state'] = 'disabled'
    else:
        btn00['state'] = 'normal'
        btn01['state'] = 'normal'
        btn02['state'] = 'normal'
        btn10['state'] = 'normal'
        btn11['state'] = 'normal'
        btn12['state'] = 'normal'
        btn20['state'] = 'normal'
```

```
        btn21['state'] = 'normal'
        btn22['state'] = 'normal'
        btn00['text'] = ''
        btn01['text'] = ''
        btn02['text'] = ''
        btn10['text'] = ''
        btn11['text'] = ''
        btn12['text'] = ''
        btn20['text'] = ''
        btn21['text'] = ''
        btn22['text'] = ''
        symbol = 'O'
        board = [['','',''],['','',''],['','','']]

def checkresult(fil,col):
    global board
    dato = board[fil][col]
    sw   = True
    for tmp in range(3):                # Verifico si
se completó una fila
        if board[fil][tmp]!=dato:
            sw = False
            break
    if not sw:
        sw = True
        for tmp in range(3):            # Verifico si
se completó una columna
```

```
            if board[tmp][col]!=dato:
                sw = False
                break
    if not sw and fil==col:            # Es un elemen
to de la diagonal principal
        sw = True
        for tmp in range(3):
            if board[tmp][tmp]!=dato:
                sw = False
                break
    if not sw and fil+col==2:          # Es un elemen
to de la diagonal secundaria
        sw = True
        for tmp in range(3):
            if board[tmp][2-tmp]!=dato:
                sw = False
                break
    if sw:
        if board[fil][col]=='X':
            res = "Player 2 WON"
        else:
            res = "Player 1 WON"
        gamestatus(False)
    else:
        sw = True
        for ele in board:
            for dat in ele:
```

```
                if dat=='':
                    sw = False
                    break
        if sw:
            res = "Draw Game"
        else:
            res = ''
    return res

def putmark(btn,fil,col):
    global symbol
    global board
    btn['text'] = symbol
    board[fil][col] = symbol
    res=checkresult(fil,col)
    if res!='':
        messagebox.showinfo(message=res, title="Game
results")
    else:
        if symbol=='X':
            symbol ='O'
        else:
            symbol = 'X'

def btn00clk():
    putmark(btn00,0,0)
    btn00['state'] = 'disabled'
```



```
def btn01clk():
    putmark(btn01,0,1)
    btn01['state'] = 'disabled'

def btn02clk():
    putmark(btn02,0,2)
    btn02['state'] = 'disabled'

def btn10clk():
    putmark(btn10,1,0)
    btn10['state'] = 'disabled'

def btn11clk():
    putmark(btn11,1,1)
    btn11['state'] = 'disabled'

def btn12clk():
    putmark(btn12,1,2)
    btn12['state'] = 'disabled'

def btn20clk():
    putmark(btn20,2,0)
    btn20['state'] = 'disabled'

def btn21clk():
    putmark(btn21,2,1)
```



```
    btn21['state'] = 'disabled'

def btn22clk():
    putmark(btn22,2,2)
    btn22['state'] = 'disabled'

def btn30clk():
    gamestatus(True)

window = Tk()
btn00 = Button(window, text='', fg='blue', height=5,
width=10, command=btn00clk)
btn01 = Button(window, text='', fg='blue', height=5,
width=10, command=btn01clk)
btn02 = Button(window, text='', fg='blue', height=5,
width=10, command=btn02clk)
btn10 = Button(window, text='', fg='blue', height=5,
width=10, command=btn10clk)
btn11 = Button(window, text='', fg='blue', height=5,
width=10, command=btn11clk)
btn12 = Button(window, text='', fg='blue', height=5,
width=10, command=btn12clk)
btn20 = Button(window, text='', fg='blue', height=5,
width=10, command=btn20clk)
btn21 = Button(window, text='', fg='blue', height=5,
width=10, command=btn21clk)
```

```
btn22 = Button(window, text='', fg='blue', height=5,
width=10, command=btn22clk)
btn30 = Button(window, text='New game', fg='blue', he
ight=3, width=15, command=btn30clk)
#btn22.bind('<Button-1>',btn22clk)
btn00.place(x=25,y=20)
btn01.place(x=105,y=20)
btn02.place(x=185,y=20)
btn10.place(x=25,y=106)
btn11.place(x=105,y=106)
btn12.place(x=185,y=106)
btn20.place(x=25,y=192)
btn21.place(x=105,y=192)
btn22.place(x=185,y=192)
btn30.place(x=150,y=280)
window.title('TicTacToe V1.0')
# Se usa para centrar la ventana
vw = 290
vh = 350
vx = window.winfo_screenwidth() // 2 - vw // 2
vy = window.winfo_screenheight() // 2 - vh // 2
posicion = str(vw) + "x" + str(vh) + "+" + str(vx) +
"+" + str(vy)
window.geometry(posicion)
window.resizable(0,0)
window.mainloop()
```

JUEGO DEL AHORCADO – QUEMADO (HANGMAN)

```
import os
from random import randint
palabra   = ""
correctas = []
errores   = []
rpta      = 'si'
partes    = ['''
   +-----+
         |
         |
         |
         |
         |
     ========
''', '''
   +-----+
   0     |
         |
         |
         |
         |
     ========
''', '''
   +-----+
   0     |
   |     |
         |
```



```
''', '''
   +-----+
   0     |
  /|\    |
  / \    |
         |
         |
     ========
''']
palabras  = ['software', 'windows', 'computador', 'an
álisis', 'diseño', 'trabajo', 'colibrí', 'cantinflas'
, 'camión']

def nuevapalabra(lista) -> str:
    """ Delecciona de forma aleatoria una palabra de
la lista """
    try:
        palabra = lista[randint(0, len(lista)-1)]
    except Exception:
        palabra = 'bailarina'
    return palabra

def cambio(letra) -> str:
    """ Ignora las tildes, retornando la letra sin ti
lde """
    try:
        if letra =='á':
```

```
            letra = 'a'
        elif letra =='é':
            letra = 'e'
        elif letra=='í':
            letra = 'i'
        elif letra=='ó':
            letra = 'o'
        elif letra in 'úü':
            letra = 'u'
    except Exception:
        pass
    return letra

def mostrar(palabra, correctas, errores) -> None:
    """ Despliega el estado del juego """
    try:
        print("AHORCADO")
        print(partes[len(errores)])
        res = 'Intentos fallidos: '
        for letra in errores:
            res = res+f'{letra} '
        print(res)
        enblanco = ""
        for letra in palabra:
            if cambio(letra) in correctas:
                enblanco = enblanco+letra
            else:
```

```
                enblanco = enblanco+"_"
        print(enblanco)
    except Exception:
        pass

def leer(entradas) -> str:
    """ Toma una letra de la entrada y realiza las va
lidaciones necesarias """
    sw = True
    try:
        while sw:
            entrada = input("Ingrese una letra: ").st
rip().lower()
            if len(entrada) != 1:
                print("Por favor escriba una sola let
ra.")
            elif entrada in entradas:
                print("Letra repetida, ya fue utiliza
da.")
            elif entrada not in "abcdefghijklmnñopqrs
tuvwxyz":
                print("Debe indicar una letra.")
            else:
                sw = False
    except Exception:
        pass
    return entrada
```

```
def cls() -> None:
    """ Se encarga de borrar la pantalla dependiendo
del sistema operativo """
    if os.name == 'nt':
        os.system('cls')
    else:
        os.system('clear')

def valida(letra,palabra) -> bool:
    """ Se encarga de verificar si una letra se encue
ntra en la palabra, ignora las tildes """
    try:
        sw = False
        if letra=='a' and ('a' in palabra or 'á' in p
alabra):
            sw = True
        elif letra=='e' and ('e' in palabra or 'é' in
 palabra):
            sw = True
        elif letra=='i' and ('i' in palabra or 'í' in
 palabra):
            sw = True
        elif letra=='o' and ('o' in palabra or 'ó' in
 palabra):
            sw = True
```

```
        elif letra=='u' and ('u' in palabra or 'ú' in
 palabra or 'ü' in palabra):
            sw = True
        else:
            sw = letra in palabra
    except Exception:
        pass
    return sw

def iniciar() -> None:
    """ Inicializa las variables del juego """
    global palabra
    global correctas
    global errores
    global rpta
    palabra = nuevapalabra(palabras)
    correctas = []
    errores = []
    rpta = 'si'
    cls()
    mostrar(palabra,correctas,errores)

iniciar()
while rpta.startswith('s'):
    letraEntrada = leer(errores+correctas)
    if valida(letraEntrada,palabra):
        correctas.append(letraEntrada)
```

```
    else:
        errores.append(letraEntrada)
    cls()
    mostrar(palabra, correctas, errores)
    todas = True
    for letra in palabra:
        if cambio(letra) not in correctas:
            todas = False
            break
    if todas or len(errores) == len(partes)-1:
        if todas:
            print("Excelente, GANASTE.")
        else:
            print("Lo siento, PERDISTE.")
        print(f"Intentos perdidos  : {len(errores)}")
        print(f"Intentos correctos : {len(correctas)}
")
        print(f"La palabra era     : {palabra}")
        rpta = input("Desea jugar otra partida [Si/No
] : ").lower()
        if rpta.startswith('s'):
            iniciar()
```