<codoa codo/>

Codo a Codo inicial Python

Trabajo Integrador Final

BA

Trabajo integrador final - Objetivo

El objetivo del *Trabajo integrador final* es aplicar todo lo aprendido durante el curso y aportar lo que consideres conveniente para el desarrollo del mismo.

La entrega del *Trabajo integrador final* es de carácter obligatorio.

El aporte de contenido extracurricular (no contemplado en el temario del curso) es opcional.

Consigna

Deberás crear una aplicación de consola símil “Inicio de sesión”, que permita:

- Crear usuari@s y sus contraseñas: **los usuari@s deben únicos**.
- Iniciar sesión de usuari@: debe iniciar con su correspondiente contraseña y dar la alternativa de **3 (tres) intentos** para ingresar la contraseña correcta.
- Salir del programa: el programa **finalizará cuando lo desee el usuari@**, haya iniciado sesión o no.
- **Almacenar en un archivo los usuarios y sus contraseñas (persistencia de datos)**.

A continuación detallaremos ejemplos de ejecución, almacenamiento y presentación.

Ejemplo de ejecución – Creación de usuario: Caso 1

1 – Pantalla principal de la aplicación.

```
    ** Menú **
    Sesión sin iniciar
    1 - Crear usuario
    2 - Iniciar sesión
    3 - Salir

Opción >>>
```

2 – Creación de usuario exitosa.

```
    ** Menú **
    Sesión sin iniciar
    1 - Crear usuario
    2 - Iniciar sesión
    3 - Salir

Opción >>> 1
Ingrese usuario: UsuarioDePrueba
Ingrese contraseña:
Usuario creado exitosamente!
ENTER para continuar >>>
```

Ejemplo de ejecución – Creación de usuario: Caso 2

1 – Pantalla principal de la aplicación.

```
    ** Menú **
    Sesión sin iniciar
    1 - Crear usuario
    2 - Iniciar sesión
    3 - Salir

Opción >>>
```

2 – Creación de usuario incorrecta (usuario ya existente).

```
    ** Menú **
    Hola UsuarioDePrueba!
    1 - Crear usuario
    2 - Iniciar sesión
    3 - Salir

Opción >>> 1
Ingrese usuario: UsuarioDePrueba
¡El usuario ya existe!
ENTER para continuar >>>
```

Ejemplo de ejecución – Inicio de sesión: Caso 1

1 – Pantalla principal de la aplicación.

```
    ** Menú **
    Sesión sin iniciar
    1 - Crear usuario
    2 - Iniciar sesión
    3 - Salir

Opción >>>
```

2 – Inicio de sesión correcto.

```
    ** Menú **
    Sesión sin iniciar
    1 - Crear usuario
    2 - Iniciar sesión
    3 - Salir

Opción >>> 2
Ingrese usuario: UsuarioDePrueba
Ingrese contraseña (3 intentos):
¡Ingreso exitoso!
ENTER para continuar >>>
```

Ejemplo de ejecución – Inicio de sesión: Caso 2

1 – Pantalla principal de la aplicación.

```
    ** Menú **
    Sesión sin iniciar
    1 - Crear usuario
    2 - Iniciar sesión
    3 - Salir

Opción >>>
```

2 – Inicio de sesión incorrecto.

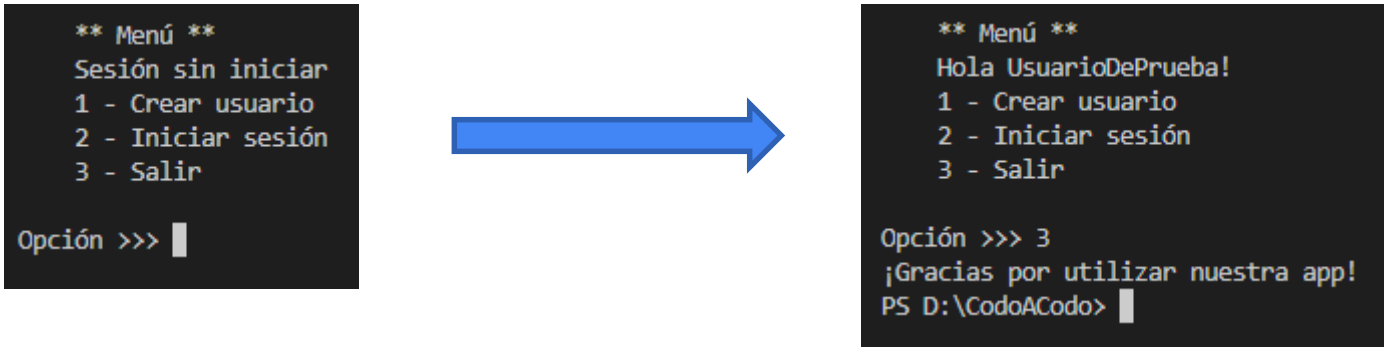
```
    ** Menú **
    Sesión sin iniciar
    1 - Crear usuario
    2 - Iniciar sesión
    3 - Salir

Opción >>> 2
Ingrese usuario: UsuarioDePrueba
Ingrese contraseña (3 intentos):
Ingrese contraseña nuevamente (intento 2):
Ingrese contraseña nuevamente (intento 3):
¡Contraseña inválida!
ENTER para continuar >>>
```

Ejemplo de ejecución – Finalización:

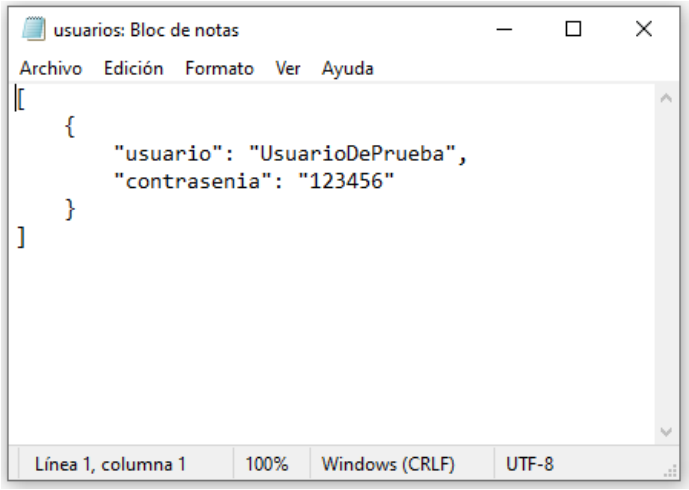
1 – Pantalla principal de la aplicación.

2 – Finaliza el programa.



Ejemplo de ejecución – Almacenamiento:

Los usuarios deberán almacenarse en un archivo con extensión “txt” y en formato JSON (JavaScript Object Notation).



Desarrollo de la aplicación:

Durante la clase los estudiantes serán orientados por sus docentes en cuanto a la resolución de la consigna planteada mediante diagramas de flujo o pseudocódigo.
La actividad podrá ser realizada de manera individual o por grupos de no más de 2 (dos) integrantes y sujeto al criterio de cada docente.

Presentación:

La presentación deberá contener los programas escritos por el/l@s estudiantes junto al archivo que almacena los datos y podrá realizarse:

- por link de Google Drive ó
- por link a Repositorio de GitHub.

En ambos casos deberán ser cuentas registradas a nombre de los estudiantes.

Criterios de evaluación:

Se considerarán los siguientes ítems para evaluar el Trabajo Integrador Final:

- Cumplimiento completo de la consigna
- Programa/s sin errores de ejecución
- **Aplicación de "buenas prácticas" en cuanto al código del programa: convenciones con respecto a nombres de variables y funciones, indentados, comentarios (o *docstrings*) y claridad en la lectura del código.** Con el foco puesto en una futura salida laboral, recordar que un desarrollador que se integra a un equipo de trabajo *lee más código del que produce*.

Aportes adicionales (opcional):

Para enriquecer y agregar valor al trabajo, se proponen los siguientes puntos:

1 - Verificación de longitud de contraseña: corroborar que la misma tenga una extensión mínima y máxima. Por ejemplo: un mínimo de 6 y un máximo de 10 caracteres, que deberán mencionarse en la interfaz de usuario.

2 - Verificación en creación de contraseña: esto quiere decir que al crearla, se solicitará una segunda vez la misma y deberá coincidir el primer ingreso para la creación del usuario.

3 - Persistencia de datos en base de datos: en lugar de almacenar los datos en un archivo de extensión “txt”, se podrán almacenar en una base de datos relacional (SQL).

4 - Interfaz gráfica: en lugar de ejecución por consola, implementar paquetes de interfaz gráfica, por ejemplo: Tkinter.

Aportes adicionales - Aclaración importante:

Los puntos mencionados en la página anterior, sólo podrán implementarse de manera secuencial. Es decir, por ejemplo, que no se permitirá la persistencia de datos en BBDD sin la verificación de longitud de contraseña y doble ingreso para su creación.

Es importante tener en cuenta esto, ya que la finalidad del curso radica en el aprendizaje de los fundamentos de la programación, comprendido por la interpretación de un problema y la implementación de algoritmos para la resolución del mismo. Consideramos que con éstas herramientas fundamentales, se consolida de mejor manera el aprendizaje.

Por éste motivo los "*Aportes adicionales"* son opcionales, no obstante te invitamos a desafiarte e implementarlos, aunque sea de manera posterior a la presentación del trabajo.

¡Muchos éxitos y a enfrentar el desafío!

…
Lo simple es mejor que lo complejo.
Complejo es mejor que complicado.
Plano es mejor que anidado.
Disperso es mejor que denso.
La legibilidad cuenta.
… (*)

* Fragmento del Zen de Python - http://www.thezenofpython.com/

Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.

Todo en el Aula Virtual.