TRABAJO PRACTICO

ALGORITMOS Y ESTRUCTURAS

DE DATOS I

DOCUMENTACIÓN

UNITURNOS SYS GSJ

Grupo 4:

Veron Peralta Franco Tomas - LU1201623

Profesor:

Maria Eugenia Varando

Cuatrimestre: 2 – Año: 2024

UNIVERSIDAD ARGENTINA DE LA EMPRESA

FACULTAD DE INGENIERÍA Y CIENCIAS EXACTAS

Índice de Contenidos

1. Instrucciones de utilización

El sistema de gestión de turnos está diseñado para ser ejecutado desde la línea de comandos. A continuación, se describen los pasos y prerrequisitos para usar el sistema, junto con ejemplos de comandos y formatos de reportes.

1.1. Prerrequisitos

- Contar con acceso a una terminal de comandos.
- Python 3.0 o superior.
- Conocimientos básicos de terminal de comandos

1.2. Pasos de utilización del programa

1) Inicialización del programa

- Abra una terminal de comandos.
- Muévase hacia la ubicación donde tenga guardado el programa
- Si dentro del programa no encuentra la carpeta “venv” ejecute:
  python -m venv venv
- Active el entorno virtual con:
  ./venv/scripts/Activate
- Ejecute el archivo de inicio “main.py”
  python main.py en Windows
  python3 main.py en macOS y Linux

2) Carga de Operaciones

PRIMER INICIO:

Al ejecutar el programa por primera vez, se indicara en pantalla que ocurrió un error al cargar el archivo “operaciones.json”, lo que dará paso a la carga de operaciones.

Comenzada la carga de operaciones se le pedirá al usuario que ingrese un código de operación, el cual será un valor entero entre 100 y 999, posterior a esto, el programa le pedirá al usuario que ingrese el nombre de la operación relacionada con el código introducido anteriormente.

El programa continuara exigiendo el ingreso de operaciones hasta alcanzar 5 operaciones validas.

Una vez cargadas las 5 operaciones, dentro de la carpeta “files” se creara un archivo “operaciones.json” el cual contendrá dichas operaciones y se pasara a la carga de turnos en el sistema

PROXIMOS INICIOS

En los próximos inicios, el programa recuperará las operaciones del archivo “operaciones.json” dentro de la carpeta “files” y pasara a la carga de turnos. En caso de que ocurra un error con la carga del archivo o este se encuentre inconsistente, el programa comenzara la carga de nuevas operaciones, explicada en PRIMER INICIO.

3) Carga de turnos

INGRESO DE DATOS PERSONALES

Iniciada la carga de turnos lo primero con lo que se procederá es con el ingreso de datos personales del alumno, donde deberá indicar su legajo, posteriormente se le exigirá el ingreso de nombre, apellido y mail. Posteriormente se pasará a la etapa de ELECCIÓN DE OPERACIÓN E INFORME DE PUESTO DE ATENCIÓN.

Si el alumno ya hubiese hecho consultas anteriores, se le pedirá su legajo únicamente y sus datos serán cargados automáticamente dando paso a la ELECCIÓN DE OPERACIÓN E INFORME DE PUESTO DE ATENCIÓN

ELECCIÓN DE OPERACIÓN E INFORME DE PUESTO DE ATENCIÓN

En esta etapa se le mostrará al alumno una tabla con todas las operaciones disponibles para consultar indicando código y nombre de operación.

El alumno podrá seleccionar una operación válida ingresando el código o nombre de esta. Posteriormente se le indicara el puesto al que debe dirigirse a realizar su consulta

VALORACION DE CONFORMIDAD Y ESTADO

Finalizada la consulta del alumno le pedirá que indique del 1 al 3, que tan conforme se encuentra con la atención, siendo:

- 1: Inconforme
- 2: Regular
- 3: Conforme

Posterior al ingreso de la conformidad, se le pedirá al alumno utilizando también una valoración del 1 al 3, indicando el estado de su consulta, siendo:

- 1: No solucionado
- 2: En revisión
- 3: Pendiente

Posterior a esto finalizará la consulta, la cual dará paso a una nueva pidiendo el ingreso de un nuevo legajo (ver INGRESO DE DATOS PERSONALES). En caso del legajo ser 12000 se finalizará la carga de turnos y se pasará a la generación de informes.

4) Generación de informes y archivos

En esta etapa se mostrará una tabla en la que se indicará cantidad de tramites, cantidad y porcentajes de las valoraciones tanto de conformidad como de estado y datos más relevantes de cada consulta recibida.

También se genera una serie de archivos CSV para un posterior análisis mediante gráficos. Habiendo ocurrido esto el programa habrá finalizado.

5) Archivos

Al finalizar la ejecución del programa se obtendrá dentro de la carpeta “files” tres archivos CSV y uno JSON

- alumnos.csv
- turnos.csv
- operaciones.csv
- operaciones.json

alumnos.csv: Archivo en el que se contendrá una lista de los datos personales de los alumnos que realizaron consultas. Se presentará con el siguiente formato:

id de alumno, nombre, apellido, legajo, email

turnos.csv: Archivo en el que se contendrá una lista de los datos de cada turno realizada, se presentará en el siguiente formato:

id del turno, id del alumno, puesto de atención, valoración de conformidad, valoración de estado, id de operación, fecha, hora

operaciones.csv: Archivo en el que se contendrá una lista de las operaciones disponibles para cada consulta, se presentará en el siguiente formato:

id de operación, código de operación, nombre de operación

operaciones.json: Archivo en el que se contendrá una lista de las operaciones disponibles para cada consulta, el cual será cargado al inicio del sistema.

6) Anotaciones

- Si se desea cargar nuevas operaciones, se deberá eliminar el archivo “operaciones.json” de la carpeta “files” .
- Cada ingreso de un dato que deba realizarse cuenta con sus respectivas validaciones.

2. Manual del desarrollador

El código está organizado en varios módulos para facilitar su mantenimiento y extensibilidad. A continuación, se describe la estructura y las principales funciones del código.

2.1. Estructura del código

```
C:.
|   main.py
|
├───app
|   |   carga_operaciones.py
|   |   carga_turnos.py
|   |   generar_informes.py
|   |   ingreso_operacion.py
|   |   ingreso_usuario.py
|   |   validar_operaciones.py
|   |
|   ├───informes
|   |   |   informe_general.py
|   |   |   __init__.py
|   |
|   ├───utils
|   |   |   archivos.py
|   |   |   busqueda.py
|   |   |   calcular_porcentaje.py
|   |   |   contar_cantidades.py
|   |   |   expresiones_regulares.py
|   |   |   hora_random.py
|   |   |   ingreso_dato.py
|   |   |   mostrar_tabla_operaciones.py
|   |   |   validaciones_regex.py
|   |   |   validar_rango.py
|   |   |   __init__.py
|
└───files
|   |   |   alumnos.csv
|   |   |   operaciones.json
|   |   |   turnos.csv
```

2.2. Módulos y funciones

ESTRUCTURAS BÁSICAS

IAlumno {
id: int,
nombre: str,
apellido: str,
legajo: int,
email: str
}

ITurno {
id: int,
idAlumno: int,
puestoAtención: int,
conformidad: int,
estado: int,
idOperación: int,
fecha: str,
hora: str
}

IOperacion {

id:int,
codigo: int,
nombre: string }

}

FUNCIONES PRINCIPALES

1) Main

Es el punto de entrada de la aplicación donde se ejecutan las funciones principales.
Nombre: main
Path: ~/main.py

2) Carga Operaciones

Función que se encarga de cargar las operaciones a consultar en el sistema de un archivo json, de fallar los carga manualmente.
Nombre: cargaOperaciones
Path: ~/app/carga_operaciones.py
Parámetros:

- ops: list < IOperacion >
- cantidadOperaciones: int

3) Carga Operaciones Manual

Función que se encarga de cargar operaciones en una lista, de forma manual.
Nombre: cargaOperacionesManual
Path: ~/app/carga_operaciones.py
Parámetros:

- ops: list < IOperacion >
- cantidadOperaciones: int

Depende de: Carga Operaciones

4) Carga Turnos

Función que se encarga de tomar los turnos de cada alumno.
Nombre: cargaTurnos
Path: ~/app/carga_turnos.py
Parámetros:

- liOps: list < IOperacion >
- liAlumnos: list < IAlumno >
- liTurnos: list < ITurno >

5) Ingreso Usuario

Función que gestiona el ingreso de los datos personales del alumno.
Nombre: ingresoUsuario
Path: ~/app/ingreso_usuario.py
Parámetros:

- legajo: int
- liAlumnos: list < IAlumno >

Retorna: IAlumno

Depende de: Carga Turnos

6) Ingreso Operación

Función que gestiona la selección de la operación a consultar en la carga del turno.
Nombre: ingresoOperacion
Path: ~/app/ingreso_operacion.py
Parámetros:

- liOps: list < IOperacion >

Retorna:int
Depende de: Carga Turnos

7) Generar Informes

Función encargada de generar las tablas de informes y los archivos donde se almacenan las consultas registradas.
Nombre: generarInformes
Path: ~/app/generar_informes.py
Parámetros:

- liOp: list < IOperacion >
- liAlumnos: list < IAlumno >
- liTurnos: list <ITurno>

FUNCIONES UTILES

1) Ingreso Dato

Función encarga de recibir el ingreso de un dato en formato texto por consola.
Nombre: ingresoDato
Path: ~/app/utils/ingreso_dato.py
Parámetros:

- mensaje: str

Retorna: str

2) Ingreso Dato Int

Función encargada de recibir un valor de numero entero por consola entre un determinado rango de número.
Nombre: ingresoDatoInt
Path: ~/app/utils/ingreso_dato.py
Parámetros:

- minimo: int
- maximo: int
- mensajeIngreso: str

Retorna: str

3) Contar Cantidades Listas

Función utilizada para contar las ocurrencias de valoraciones de conformidad y estado
Nombre: contarCantidadesLista.
Path: ~/app/utils/contar_cantidades.py
Parámetros:

- liTurnos: list< ITurno >
- listaCantidades: list< int >

4) Mostrar Tabla Operaciones

Función encarga de mostrar la tabla de operaciones cargadas en el sistema.
Nombre: mostrarTablaOperaciones
Path: ~/app/utils/mostrar_tabla_operaciones
Parámetros:

- liOps: list < IOperacion >

5) Validar Email

Función encargada de validar que una cadena de texto cumpa que formato esperado de un email.
Nombre: validarEmail
Path: ~/app/utils/validaciones_regex.py
Parámetros:

- email: str

Retorna: bool

6) Validar Nombre

Función encargada de validar que una cadena de texto cumpa que formato esperado de un nombre.
Nombre: validarNombre
Path: ~/app/utils/validaciones_regex.py
Parámetros:

- nombre: str

Retorna: bool

7) Validar Entero

Función encargada de validar que una cadena de texto cumpa que formato esperado de un numero entero.
Nombre: validarEntero
Path: ~/app/utils/validaciones_regex.py
Parámetros:

- numero: str

Retorna: bool

8) Validar Rango

Función encargada de validar que un numero se encuentre dentro de un rango
Nombre: validarRango.
Path: ~/app/utils/validarRango.py
Parámetros:

- valor: int
- minimo: int
- maximo: int

Retorna: 1 | 0

9) Calcular Porcentaje

Función utilizada para calcular los porcentajes de las ocurrencias de las valoraciones de conformidad y estado.
Nombre: calcularPorcentaje
Path: ~/app/utils/calcular_porcentaje.py
Parámetros:

- total: int
- cantidades: list < int >
- porcentajes: list < int >

10) Búsqueda OBJ

Función encargada de buscar donde se encuentra un valor dentro de una lista de diccionarios.
Nombre: busquedaOBJ
Path: ~/app/utils/busqueda.py
Parámetros:

- lista: list< any >
- clave: str
- valor: int | str

Retorna: int

11) Carga CSV

Función encargada de recuperar los datos de un archivo CSV
Nombre: cargaCSV
Path: ~/app/utils/archivos.py
Parametros:

- path: str

Retorna: list < str >

12) Generar CSV

Función encargada de generar un archivo CSV a partir de ciertos datos
Nombre: generarCSV
Path: ~/app/utils/archivos.py
Parametros:

- arrTuplas: list < tuples < any > >
- path: str

Retorna: bool

13) Carga JSON

Función encargada de recuperar los datos de un archivo JSON
Nombre: cargaJSON
Path: ~/app/utils/archivos.py
Parametros:

- path: str

Retorna: list < dict >

14) Generar JSON

Función encargada de generar un archivo JSON a partir de ciertos datos
Nombre: generarJson
Path: ~/app/utils/archivos.py
Parametros:

- data: list < dict >
- path: str

15) Hora Random

Función encargada de generar una hora de forma aleatoria
Nombre: horaRandom
Path: ~/app/utils/hora_random.py
Parámetros:

- min: int
- max: int

Retorna: str