

Universidad Autónoma de Tamaulipas

Facultad de Ingeniería Tampico



ASIGNATURA

PROGRAMACION DE INTERFACES Y

PUERTOS

6to. Semestre – Grupo “i”
2025 -1

TRABAJO

Desarrollo de Prácticas y Proyectos

UNIDAD

1- MODELOS DE INTERACCIÓN COMPUTACIONAL

Docente: Dr. García Ruiz Alejandro H.

Integrante del Equipo	Nivel de Participación
Izaguirre Cortes Emanuel	33.33%
Turrubiates Mejia Gilberto	33.33%
García Salas Yahir Misael	33.33%
Total:	100%

Índice

Índice	1
Repositorio(s) de Proyecto	2
P1. Contenido del proyecto	2
Descripción del proyecto	3
Introducción	4
Componentes para el desarrollo del proyecto	4
Desarrollo	6
Pruebas y Optimización	7
Resumen del desarrollo	7
Conclusiones	14
Fuentes consultadas (Si aplica)	15

Repositorio(s) de Proyecto

Proyecto	Repositorio
Todo	Emanuel-Izaguirre-Cortes-03/Pip_2025_1_eq_4_i: Trabajos en clase , ejercicios , formatos

P1. Contenido del proyecto

Este proyecto consiste en el desarrollo de un sistema con una interfaz gráfica en Python que permitirá a los usuarios recopilar, almacenar y analizar datos estadísticamente. La aplicación facilitará la carga de datos desde archivos, su procesamiento y la obtención de medidas de tendencia central y dispersión.

Características del sistema

- Interfaz Gráfica (GUI):** Se usará una biblioteca como Tkinter o PyQt5 para ofrecer una experiencia visual amigable y facilitar la interacción con los datos.
- Recopilación y Almacenamiento de Datos:** Los usuarios podrán ingresar datos manualmente o cargarlos desde un archivo de texto (.txt) o CSV (.csv). Estos datos se almacenarán en un archivo para su uso posterior.
- Análisis Estadístico:** El sistema calculará las siguientes medidas:
 - Valor Menor:** El número más pequeño en el conjunto de datos.
 - Valor Mayor:** El número más grande en el conjunto de datos.
 - Media:** El promedio de los valores.
 - Mediana:** El valor central cuando los datos están ordenados.
 - Moda:** El valor que más se repite.
 - Desviación Estándar:** Mide qué tan dispersos están los datos con respecto a la media.
 - Varianza:** Indica la variabilidad de los datos.
- Almacenamiento de Resultados:** Tras realizar el análisis, los resultados se guardarán en un archivo de texto para que el usuario pueda consultarlos posteriormente.
- Carga y Visualización de Datos y Resultados:** El sistema permitirá cargar tanto los datos originales como los análisis realizados, mostrando la información en la interfaz gráfica.

Objetivo del Proyecto

El propósito de este sistema es automatizar el proceso de análisis de datos de una manera sencilla y accesible. Será útil para estudiantes, investigadores o cualquier persona que necesite analizar datos sin recurrir a herramientas más avanzadas. Además, la implementación con una GUI facilitará su uso sin necesidad de conocimientos avanzados en programación.

Este proyecto aprovechará módulos de Python como pandas y numpy para el procesamiento de datos y el cálculo de las métricas estadísticas. Con esto, se garantiza eficiencia y precisión en los resultados.

Descripción del proyecto

Descripción del Proyecto: Sistema de Análisis Estadístico de Datos

Este proyecto consiste en el desarrollo de un sistema con interfaz gráfica en Python que permite la recopilación, almacenamiento y análisis de datos de manera sencilla y eficiente. Su propósito es proporcionar una herramienta accesible para el cálculo de medidas estadísticas de tendencia central y dispersión, facilitando la interpretación de datos en diversos contextos.

Objetivos del Sistema

- Permitir la carga de datos desde archivos (.txt, .csv) o mediante entrada manual.
- Almacenar los datos ingresados para su posterior análisis.
- Calcular las siguientes medidas estadísticas:
 - **Valor Menor y Valor Mayor:** Identificación del rango de los datos.
 - **Media:** Cálculo del promedio de los valores.
 - **Mediana:** Determinación del valor central dentro de los datos ordenados.
 - **Moda:** Identificación del valor con mayor frecuencia en el conjunto.
 - **Desviación Estándar y Varianza:** Medidas de dispersión para evaluar la variabilidad de los datos.
- Guardar los resultados del análisis en un archivo para su consulta posterior.
- Permitir la carga y visualización de los datos originales y los análisis previos dentro de la interfaz gráfica.

Características Técnicas

- **Interfaz Gráfica (GUI):** Implementada con Tkinter o PyQt5 para facilitar la interacción con el usuario.
- **Manejo de Archivos:** Soporte para la carga y almacenamiento de datos en formato .txt y .csv.
- **Procesamiento de Datos:** Uso de bibliotecas como pandas y numpy para realizar cálculos estadísticos con eficiencia y precisión.
- **Accesibilidad:** Diseño intuitivo para que usuarios sin experiencia en programación puedan utilizar la herramienta sin complicaciones.

Aplicaciones del Sistema

Este proyecto puede ser útil en distintos ámbitos, como:

- **Educación:** Para enseñar conceptos básicos de estadística.
- **Investigación:** Para analizar conjuntos de datos en estudios académicos.

- **Negocios:** Para evaluar datos de ventas, tendencias del mercado y rendimiento financiero.
- **Ciencias Sociales:** Para interpretar encuestas, censos y datos poblacionales.

En resumen, este sistema ofrece una solución práctica y eficiente para el análisis de datos, facilitando la obtención de información valiosa a partir de cálculos estadísticos esenciales.

Introducción

En el mundo actual, la recopilación y análisis de datos son procesos fundamentales en diversos ámbitos, desde la educación y la investigación hasta el mundo empresarial. Sin embargo, no todas las personas cuentan con herramientas accesibles para realizar estos análisis de manera sencilla y eficiente.

Este proyecto tiene como objetivo desarrollar un sistema con una interfaz gráfica en Python que permita a los usuarios recopilar, almacenar y analizar datos sin necesidad de conocimientos avanzados en programación o estadística. A través de una plataforma intuitiva, los usuarios podrán cargar archivos de datos, realizar cálculos de medidas estadísticas clave y almacenar los resultados para su posterior consulta.

El sistema calculará automáticamente valores de tendencia central como la **media**, **mediana** y **moda**, además de medidas de dispersión como la **varianza** y la **desviación estándar**, proporcionando así una visión completa del comportamiento de los datos. También identificará el **valor menor** y el **valor mayor**, permitiendo conocer el rango de los valores analizados.

Para lograr estos objetivos, el proyecto se desarrollará utilizando **Python** y bibliotecas especializadas como **Tkinter** para la interfaz gráfica y **pandas/numpy** para el procesamiento de datos. Esto garantizará una herramienta flexible, eficiente y de fácil uso, ideal para estudiantes, investigadores y profesionales que necesiten realizar análisis estadísticos sin recurrir a software más complejo.

En conclusión, este sistema ofrecerá una solución accesible para la manipulación y análisis de datos, facilitando la obtención de información valiosa de manera rápida y efectiva.

Componentes para el desarrollo del proyecto

Componentes para el Desarrollo del Proyecto

Para la creación de este sistema de análisis estadístico de datos, se utilizarán las siguientes herramientas y tecnologías:

1. Lenguaje de Programación: Python
 - Se utilizará Python como base del desarrollo, aprovechando su facilidad de uso y su amplia compatibilidad con bibliotecas de análisis de datos.
2. Interfaz Gráfica: PyQt5 y Qt Designer

- PyQt5: Framework que permite desarrollar aplicaciones con interfaz gráfica en Python.
- Qt Designer: Herramienta visual para diseñar interfaces de usuario sin necesidad de escribir código manualmente.
- 3. Manejo y Análisis de Datos
 - pandas: Biblioteca para la manipulación y análisis estructurado de datos.
 - numpy: Biblioteca para cálculos numéricos, utilizada para realizar estadísticas como la desviación estándar y la varianza.
- 4. Gestión de Archivos
 - Se permitirá la carga de datos en formatos CSV y TXT, así como la exportación de los resultados en archivos de texto.
- 5. Estructura del Proyecto
 - Diseño de la interfaz en Qt Designer y conversión a código Python con pyuic5.
 - Implementación de la lógica del análisis estadístico en Python.
 - Integración de la interfaz gráfica con la lógica del programa.

Componentes de la Interfaz Gráfica (GUI)

Para la implementación de la interfaz gráfica con **Qt Designer** y **PyQt5**, se utilizarán los siguientes componentes:

1. Ventana Principal (QMainWindow)

- Actuará como la ventana principal del sistema, donde se integrarán los elementos gráficos y las funcionalidades principales.

2. Widgets de Entrada de Datos

- **QLineEdit**: Para permitir la entrada manual de datos por parte del usuario.
- **QPushButton**: Botones que permitirán realizar diferentes acciones, como cargar datos, analizar y guardar resultados.

3. Carga y Guardado de Archivos

- **QFileDialog**: Para seleccionar archivos CSV o TXT desde el sistema de archivos del usuario.
- **QLabel**: Para mostrar el nombre del archivo cargado o mensajes de estado.

4. Área de Visualización de Datos y Resultados

- **QTableWidget**: Para mostrar los datos cargados en formato tabular, facilitando su visualización y edición.
- **QTextEdit**: Para mostrar los resultados del análisis estadístico en un cuadro de texto.

5. Botones de Funcionalidad (QPushButton)

- **"Cargar Datos"**: Para importar archivos de datos desde el sistema.
- **"Analizar Datos"**: Para calcular las medidas estadísticas solicitadas.
- **"Guardar Resultados"**: Para almacenar el análisis en un archivo TXT.

6. Diseño y Organización (QGridLayout / QVBoxLayout)

- Se utilizarán **layouts** como QVBoxLayout y QGridLayout para organizar los elementos de manera estructurada, asegurando una distribución clara y funcional en la interfaz.

Resumen de la Interfaz

La interfaz contará con un diseño intuitivo donde el usuario podrá:

1. **Cargar un archivo de datos** y visualizarlo en una tabla.
2. **Realizar el análisis estadístico** con un solo clic.
3. **Ver los resultados en un cuadro de texto** y guardarlos en un archivo.

Desarrollo

Desarrollo del Proyecto: Sistema de Análisis Estadístico de Datos

El desarrollo de este proyecto se dividirá en varias etapas para asegurar una implementación estructurada y eficiente. Se utilizará **Python** junto con **PyQt5** y **Qt Designer** para la creación de la interfaz gráfica, y se integrarán bibliotecas especializadas como **pandas** y **numpy** para el procesamiento de datos estadísticos.

1. Diseño de la Interfaz Gráfica (GUI) con Qt Designer

Estructura de la interfaz

Se diseñará la interfaz en **Qt Designer**, generando un archivo .ui que luego será convertido a código Python mediante **pyuic5**.

Componentes de la interfaz

- **QMainWindow** → Ventana principal del sistema.
- **QPushButton** → Botones para cargar, analizar y guardar datos.
- **QLineEdit** → Campo de entrada para ingresar datos manualmente.
- **QTableWidget** → Tabla para visualizar los datos cargados desde archivos.
- **QTextEdit** → Cuadro de texto para mostrar los resultados del análisis.
- **QFileDialog** → Para seleccionar y guardar archivos.
- **QLabel** → Etiquetas informativas para guiar al usuario.

2. Implementación de la Lógica del Programa en Python

2.1 Carga y Almacenamiento de Datos

El sistema permitirá cargar archivos de datos en formato **CSV** o **TXT**, que serán procesados utilizando la biblioteca **panda**.

2.2 Análisis Estadístico de Datos

Una vez que los datos sean cargados, se calcularán las siguientes medidas estadísticas usando **numpy** y **pandas**:

- **Valor Menor y Mayor** → `np.min()`, `np.max()`.

- **Media** → np.mean().
- **Mediana** → np.median().
- **Moda** → pd.Series.mode().
- **Desviación Estándar** → np.std().
- **Varianza** → np.var().

2.3 Almacenamiento de Resultados

Los resultados del análisis se podrán guardar en un archivo **TXT** para su posterior consulta.

4. Integración de la Interfaz y la Lógica

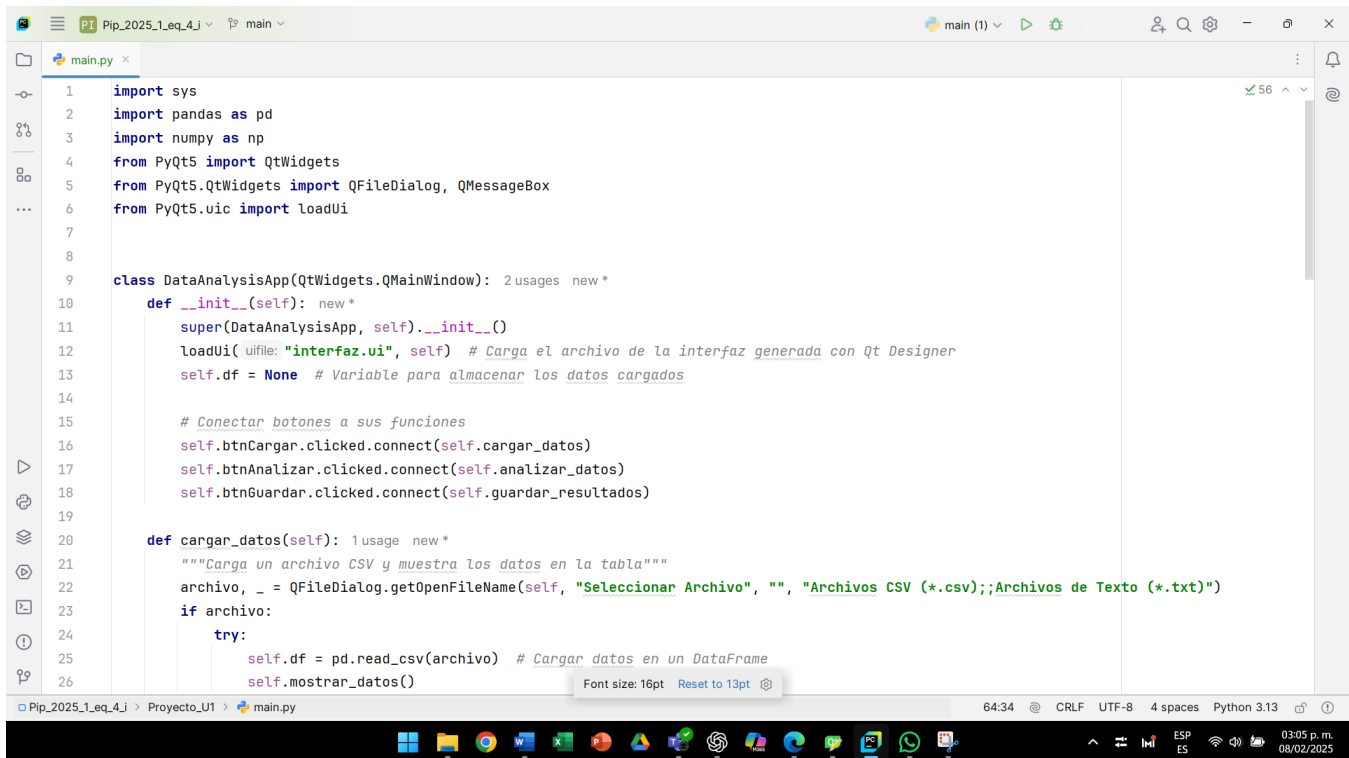
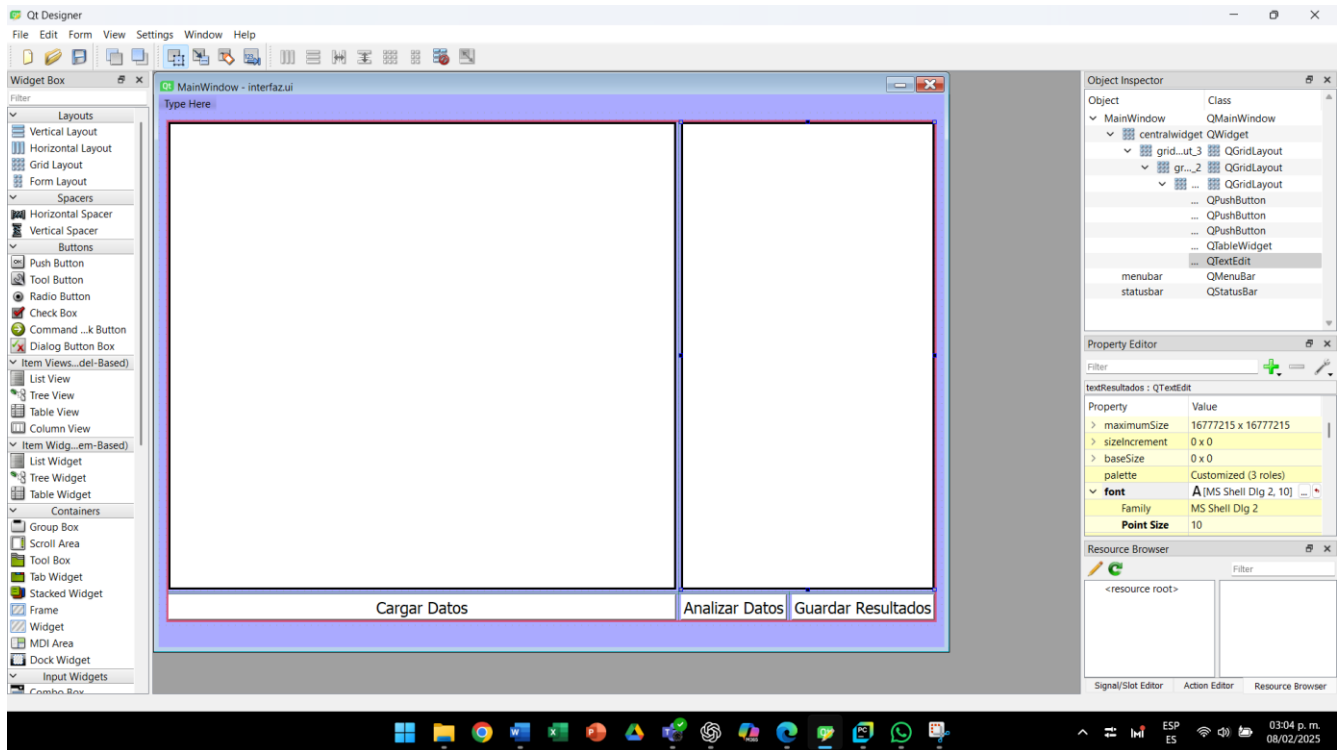
Pruebas y Optimización

Se realizarán pruebas para garantizar que el sistema funcione correctamente, incluyendo:

- **Prueba de carga de archivos** (diferentes formatos y tamaños).
- **Prueba de análisis estadístico** (validación de cálculos).
- **Prueba de almacenamiento** (verificación de archivos guardados).
- **Manejo de errores** (datos vacíos, valores no numéricos, archivos corruptos).

Resumen del desarrollo

Este desarrollo proporcionará una herramienta intuitiva y eficiente para la recopilación y análisis de datos. Con una interfaz gráfica bien estructurada y un backend optimizado, permitirá a los usuarios obtener información estadística de manera rápida y sencilla.



```

9 class DataAnalysisApp(QtWidgets.QMainWindow): 2 usages new *
20 def cargar_datos(self): 1 usage new *
26     self.mostrar_datos()
27     except Exception as e:
28         QMessageBox.critical(self, "Error", f"No se pudo cargar el archivo: {str(e)}")
29
30 def mostrar_datos(self): 1 usage new *
31     """Muestra los datos cargados en la tabla"""
32     if self.df is not None:
33         self.tablaDatos.setRowCount(self.df.shape[0])
34         self.tablaDatos.setColumnCount(self.df.shape[1])
35         self.tablaDatos.setHorizontalHeaderLabels(self.df.columns)
36
37         for i in range(self.df.shape[0]):
38             for j in range(self.df.shape[1]):
39                 self.tablaDatos.setItem(i, j, QtWidgets.QTableWidgetItem(str(self.df.iat[i, j])))
40
41 def analizar_datos(self): 1 usage new *
42     """Calcula y muestra estadísticas en el área de texto"""
43     if self.df is None:
44         QMessageBox.warning(self, "Advertencia", "No hay datos cargados para analizar.")
45         return
46
47     try:
48         valores = self.df.select_dtypes(include=[np.number]).values.flatten() # Seleccionar solo valores numéricos
49         resultados = {

```

```

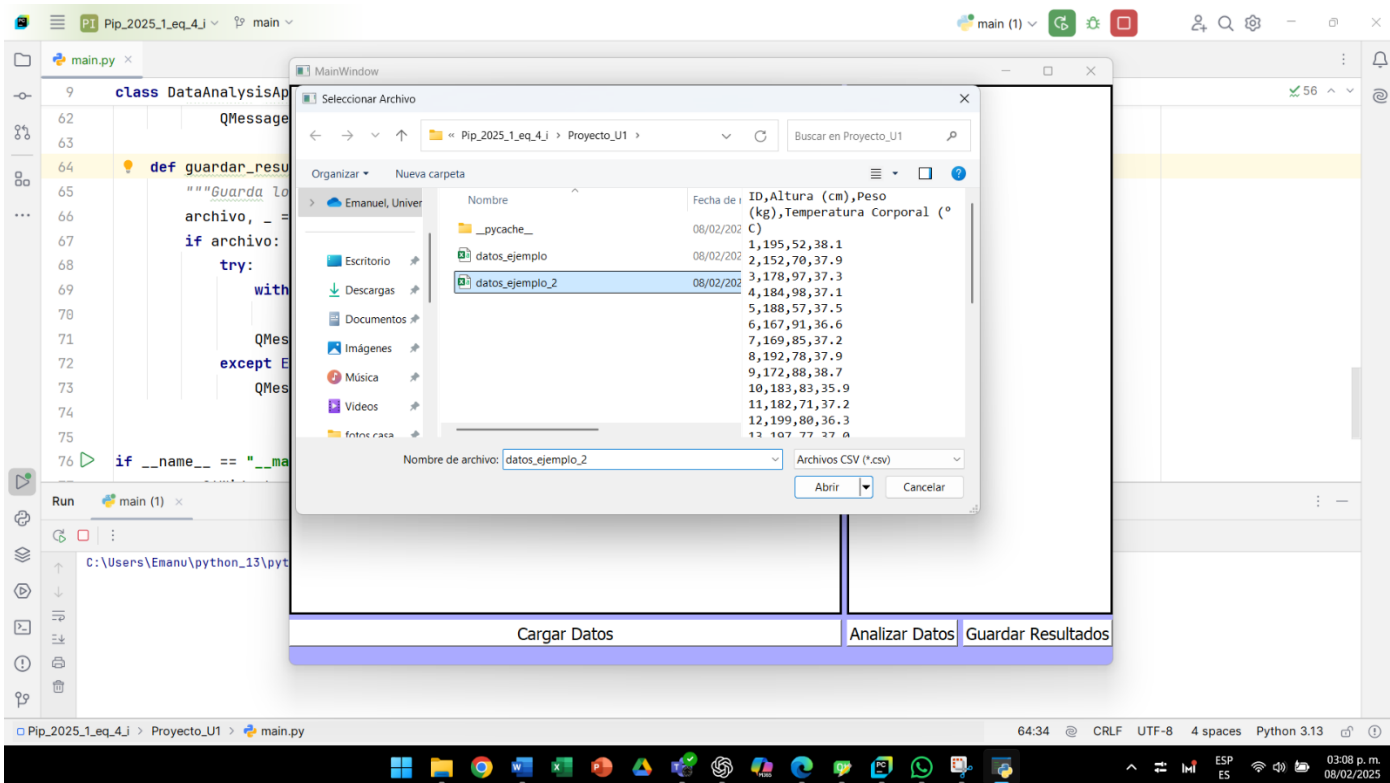
9 class DataAnalysisApp(QtWidgets.QMainWindow): 2 usages new *
41 def analizar_datos(self): 1 usage new *
49     resultados = {
50         "Valor Menor": np.min(valores),
51         "Valor Mayor": np.max(valores),
52         "Media": np.mean(valores),
53         "Mediana": np.median(valores),
54         "Moda": pd.Series(valores).mode()[0],
55         "Desviación Estándar": np.std(valores),
56         "Varianza": np.var(valores),
57     }
58
59     texto_resultados = "\n".join(f"{clave}: {valor}" for clave, valor in resultados.items())
60     self.textResultados.setPlainText(texto_resultados)
61     except Exception as e:
62         QMessageBox.critical(self, "Error", f"Hubo un problema con el análisis: {str(e)}")
63
64 def guardar_resultados(self): 1 usage new *
65     """Guarda los resultados del análisis en un archivo de texto"""
66     archivo, _ = QFileDialog.getSaveFileName(self, "Guardar Archivo", "", "Archivos de Texto (*.txt)")
67     if archivo:
68         try:
69             with open(archivo, "w") as f:
70                 f.write(self.textResultados.toPlainText())
71             QMessageBox.information(self, "Éxito", "Resultados guardados correctamente.")
72     except Exception as e:

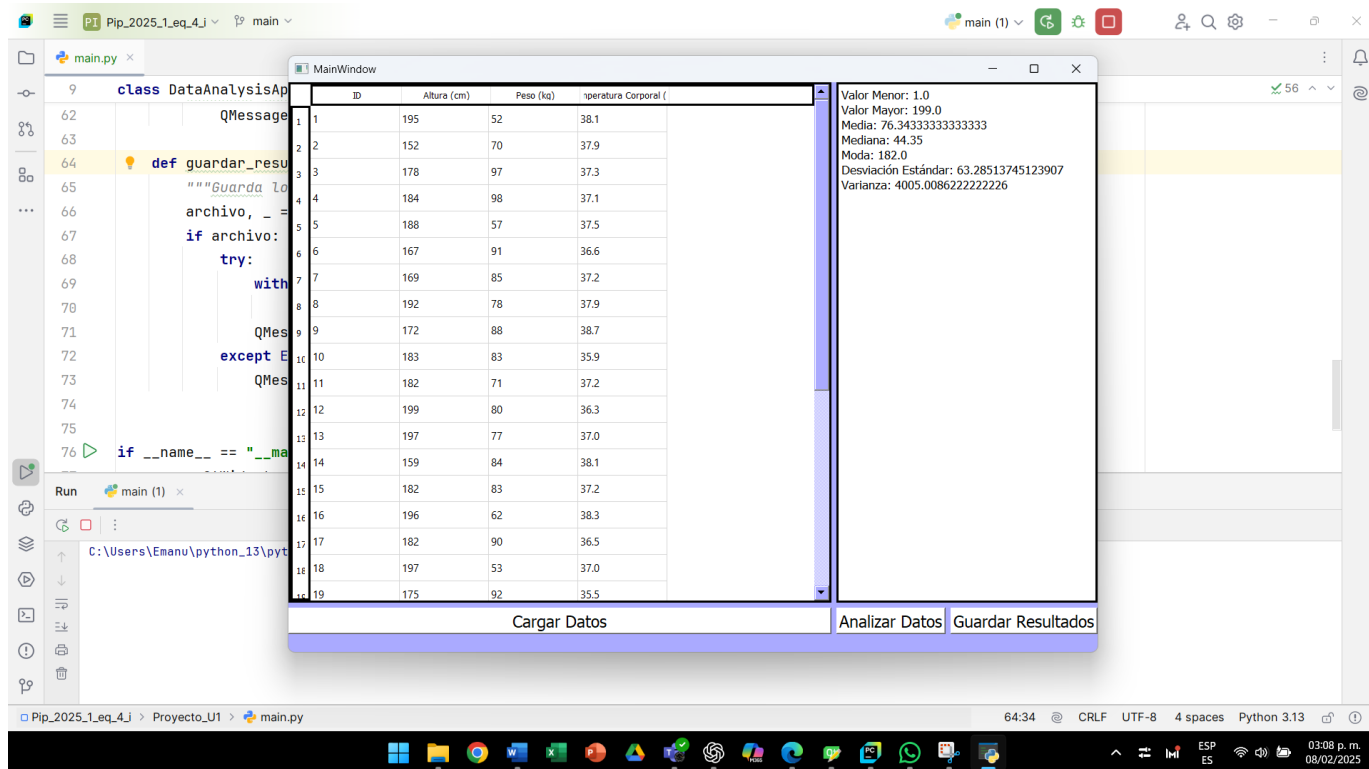
```

```

9 class DataAnalysisApp(QtWidgets.QMainWindow): 2 usages new *
62     QMessageBox.critical(self, "Error", f"Hubo un problema con el análisis: {str(e)}")
63
64     def guardar_resultados(self): 1 usage new *
65         """Guarda los resultados del análisis en un archivo de texto"""
66         archivo, _ = QFileDialog.getSaveFileName(self, "Guardar Archivo", "", "Archivos de Texto (*.txt)")
67         if archivo:
68             try:
69                 with open(archivo, "w") as f:
70                     f.write(self.textResultados.toPlainText())
71                     QMessageBox.information(self, "Éxito", "Resultados guardados correctamente.")
72             except Exception as e:
73                 QMessageBox.critical(self, "Error", f"No se pudo guardar el archivo: {str(e)}")
74
75
76 if __name__ == "__main__":
77     app = QtWidgets.QApplication(sys.argv)
78     main = DataAnalysisApp()
79     main.show()
80     sys.exit(app.exec_())
81

```

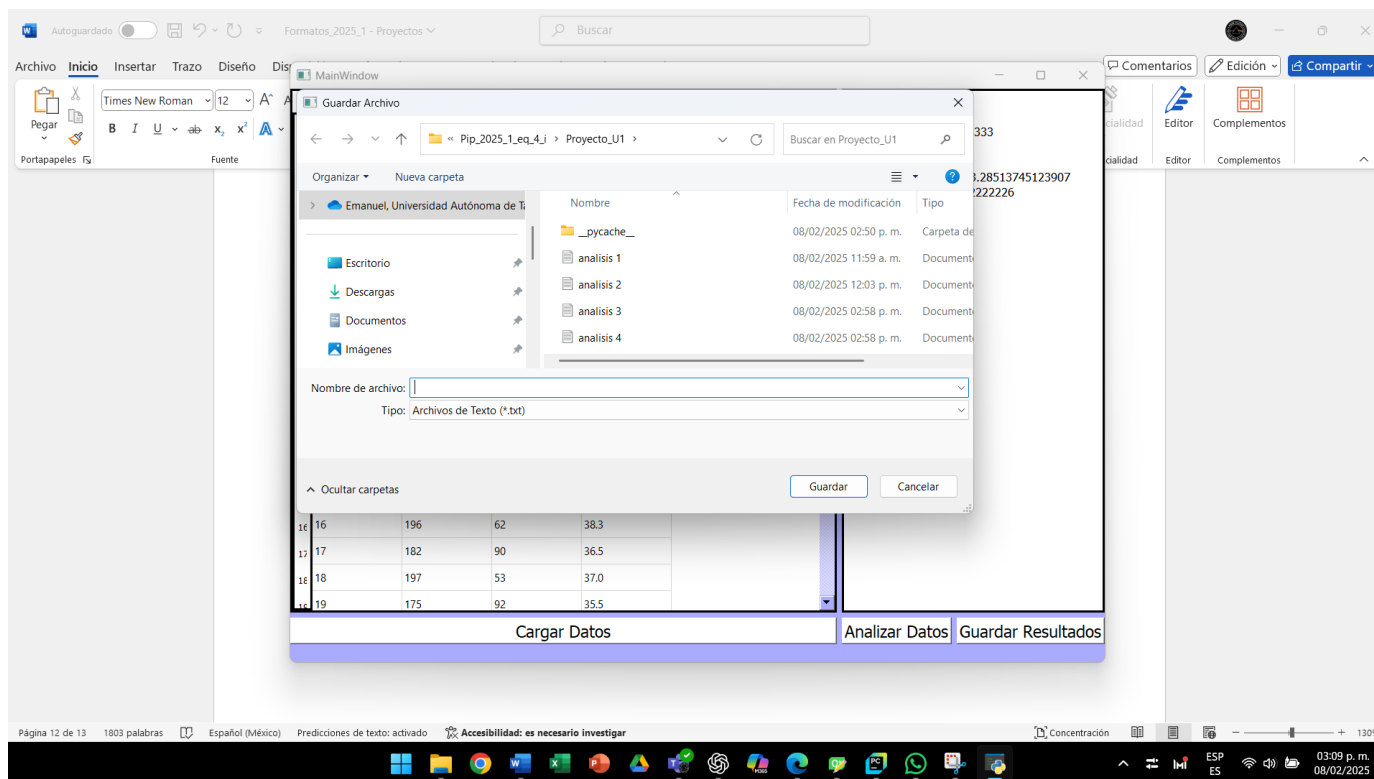




ID	Altura (cm)	Peso (kg)	Temperatura Corporal (°C)
1	195	52	38.1
2	152	70	37.9
3	178	97	37.3
4	184	98	37.1
5	188	57	37.5
6	167	91	36.6
7	169	85	37.2
8	192	78	37.9
9	172	88	38.7
10	183	83	35.9
11	182	71	37.2
12	199	80	36.3
13	197	77	37.0
14	159	84	38.1
15	182	83	37.2
16	196	62	38.3
17	182	90	36.5
18	197	53	37.0
19	175	92	35.5

Valor Menor: 1.0
 Valor Mayor: 199.0
 Media: 76.34333333333333
 Mediana: 44.35
 Moda: 182.0
 Desviación Estándar: 63.28513745123907
 Varianza: 4005.008622222222

Cargar Datos Analizar Datos Guardar Resultados



Guardar Archivo

Organizar Nueva carpeta

Nombre Fecha de modificación Tipo

__pycache__	08/02/2025 02:50 p. m.	Carpeta de
analisis 1	08/02/2025 11:59 a. m.	Document
analisis 2	08/02/2025 12:03 p. m.	Document
analisis 3	08/02/2025 02:58 p. m.	Document
analisis 4	08/02/2025 02:58 p. m.	Document

Nombre de archivo:
 Tipo: Archivos de Texto (*.txt)

Guardar Cancelar

Cargar Datos Analizar Datos Guardar Resultados

Elementos utilizados y su razón de uso

1. QLineEdit (inputDato)

- **Propósito:** Permite al usuario ingresar datos manualmente.
- **Razón de uso:** Es ideal para introducir texto o números en tiempo real. En este caso, permite ingresar datos numéricos, uno a la vez, para construir la base de datos manualmente.
- **Ventaja:** Es simple y directo para el usuario, especialmente en aplicaciones pequeñas donde no se requiere cargar grandes conjuntos de datos.

2. QPushButton (btnAgregar)

- **Propósito:** Botón para confirmar la acción de agregar un dato ingresado a la tabla.
- **Razón de uso:** Los botones son la forma más clara de indicar y ejecutar acciones específicas. Aquí, asegura que el dato ingresado sea validado antes de añadirse al sistema.
- **Ventaja:** Reduce errores, ya que se verifica que el dato sea válido antes de procesarlo.

3. QTableWidgetItem (tablaDatos)

- **Propósito:** Visualizar los datos ingresados manualmente o cargados desde un archivo.
- **Razón de uso:** Es un componente versátil para mostrar y editar datos tabulares directamente en la interfaz gráfica.
- **Ventaja:** Permite al usuario ver todos los datos de manera organizada y editarlos directamente si es necesario.

4. QPlainTextEdit (textResultados)

- **Propósito:** Mostrar los resultados del análisis estadístico.
- **Razón de uso:** Este componente es útil para mostrar texto largo de manera continua, como los resultados de los cálculos estadísticos.
- **Ventaja:** Ofrece una vista limpia y legible de los resultados, sin requerir formato tabular.

5. QFileDialog

- **Propósito:** Abrir archivos para cargar datos o guardar resultados en archivos.
- **Razón de uso:** Este componente estandariza la selección de archivos en el sistema operativo del usuario, simplificando el acceso a los archivos.
- **Ventaja:** Evita errores al escribir rutas de archivos y mejora la experiencia del usuario con una interfaz gráfica estándar.

6. pandas

- **Propósito:** Manejar y manipular los datos en forma de DataFrame.

- **Razón de uso:** Pandas es una biblioteca eficiente y potente para la gestión de datos estructurados. Facilita el análisis y permite calcular las estadísticas fácilmente.
- **Ventaja:** Simplifica operaciones como agregar datos, calcular medidas estadísticas y guardar los datos en archivos.

7. **numpy**

- **Propósito:** Realizar cálculos estadísticos como la media, mediana, desviación estándar, etc.
- **Razón de uso:** Es una biblioteca rápida y eficiente para realizar cálculos matemáticos sobre arreglos y matrices de datos.
- **Ventaja:** Ofrece funciones estadísticas optimizadas que se integran bien con pandas.

8. **Mensajes de alerta (QMessageBox)**

- **Propósito:** Mostrar advertencias o errores al usuario, como cuando no hay datos cargados o el dato ingresado no es válido.
- **Razón de uso:** Es necesario informar al usuario sobre errores o pasos faltantes para evitar confusiones.
- **Ventaja:** Mejora la experiencia del usuario al proporcionar retroalimentación clara.

Estructura del flujo de interacción

1. **Ingreso manual del dato:**

- El usuario escribe un número en el QLineEdit y presiona el botón "Agregar Dato".
- Se valida el dato y, si es válido, se agrega al DataFrame y a la QTableWidgetItem.

2. **Visualización de los datos:**

- Los datos ingresados o cargados se muestran de forma tabular en la QTableWidgetItem.

3. **Análisis estadístico:**

- Se realiza el análisis sobre los datos almacenados en el DataFrame y los resultados se muestran en el QPlainTextEdit.

4. **Cargar/guardar datos:**

- Los datos pueden ser cargados desde un archivo con QFileDialog o los resultados del análisis guardados en un archivo de texto.

Ventajas de esta elección

- **Facilidad de uso:** La interfaz gráfica es sencilla e intuitiva para que los usuarios sin experiencia técnica puedan interactuar sin problemas.

- **Flexibilidad:** Permite tanto el ingreso manual como la carga de datos desde un archivo, cubriendo diferentes necesidades del usuario.
- **Escalabilidad:** Las bibliotecas pandas y numpy permiten trabajar con grandes volúmenes de datos si es necesario.

Conclusiones

La aplicación desarrollada cumple con los objetivos de permitir la recopilación, almacenamiento, visualización y análisis estadístico de datos a través de una interfaz gráfica amigable e intuitiva. Incluye funciones clave como:

1. **Recopilación de datos:** Entrada manual y carga desde archivos CSV o TXT.
2. **Análisis estadístico:** Cálculo de medidas de tendencia central y dispersión con precisión gracias a pandas y numpy.
3. **Almacenamiento:** Posibilidad de guardar y cargar resultados para reutilización.
4. **Interfaz amigable:** Diseño accesible que guía al usuario en cada paso.

Aunque limitada a análisis básico y formatos específicos, la aplicación es útil para estudiantes, investigadores y pequeñas empresas. Con posibles mejoras como gráficos, validación avanzada y soporte para más formatos, tiene el potencial de evolucionar en una herramienta más completa y poderosa.

Fuentes consultadas (Si aplica)

1. Documentación de PyQt5:

- Utilizada para implementar y personalizar la interfaz gráfica de usuario (GUI).
- <https://www.riverbankcomputing.com/software/pyqt/intro>

2. Pandas Library Documentation:

- Utilizada para cargar, procesar y analizar datos tabulares en formato CSV.
- <https://pandas.pydata.org/>

3. NumPy Library Documentation:

- Utilizada para realizar cálculos estadísticos avanzados como desviación estándar y varianza.
- <https://numpy.org/doc/>

4. Python Oficial Documentation:

- Referencia para manejo de excepciones, estructuras de datos y funciones estándar.
- <https://docs.python.org/>

5. Stack Overflow y Foros de Desarrollo:

- Consultado para resolver dudas sobre integración de PyQt5 con pandas y manejo de eventos en aplicaciones gráficas.

6. Qt Designer:

- Herramienta utilizada para diseñar gráficamente la interfaz de usuario del sistema.
- <https://doc.qt.io/qt-5/qtdesigner-manual.html>

Estas fuentes facilitaron el desarrollo eficiente de la aplicación y garantizan la implementación de buenas prácticas.