

```
<!--Pes2025 Programacion 1-->
```

# GastroController

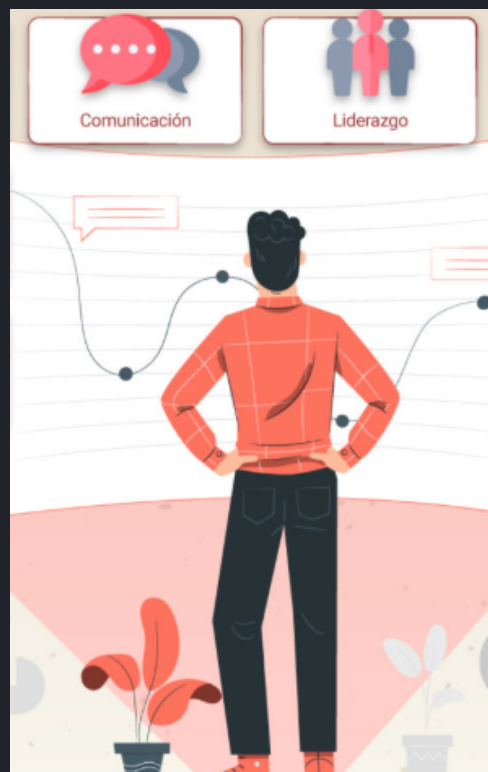
```
5000 {
```

```
<Por=["Mario Castro","Diego Chan]>
```

```
}
```



# Equipo {



MARIO CASTRO

DIRECCIÓN DE  
PROYECTO



DIEGO CHAN

DESARROLLADOR

}

# Contenidos

01

Introducción

02

Aspectos clave

03

Estructura

04

Funciones

05

Base de datos

06

Servidor

07

Planificación

# Introducción {

GastoController5000 es una aplicación que nació para suplir las necesidades del ejecutivo de hoy, su innovadora interfaz y herramientas de última generación hacen que el control de gastos sea tan fácil como un par de clicks.

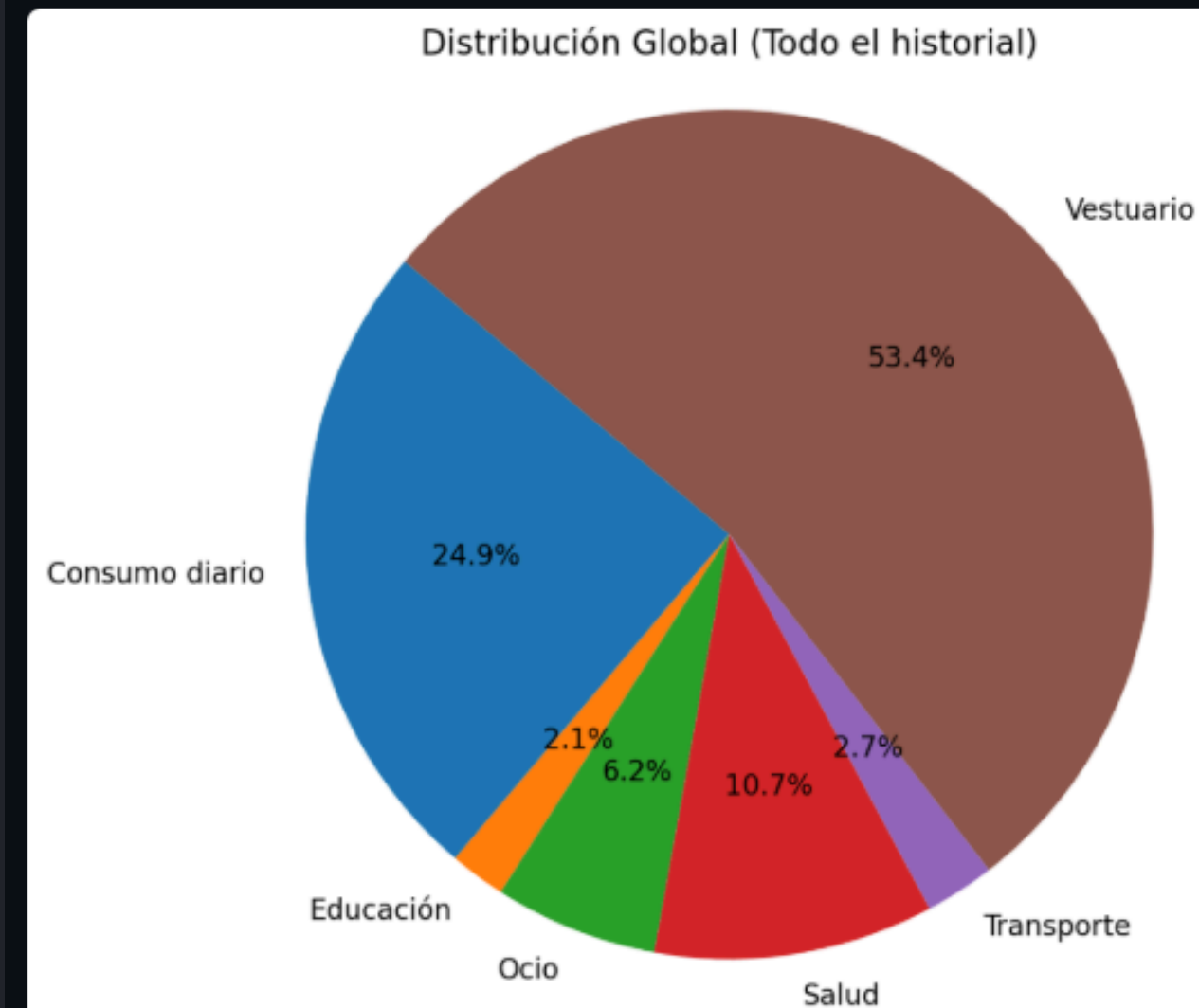
Sus funciones principales son proporcionar una proyección en el tiempo de a dónde se está dirigiendo sus gastos y dónde debería tener cuidado. De manera gráfica podrá saber siempre dónde está su dinero o a dónde fue.

## GASTOCONTROLLER5000

### + Agregar Nuevo Gasto

Producto	Categoría
<input type="text" value="Ej. Café, Gasolina..."/>	<input type="text" value="Consumo diario"/>
Monto	Fecha
<input type="text" value="0.00"/>	<input type="text" value="2025/09/26"/>
<input type="button" value="Guardar"/>	

### Distribución Total de Gastos por Categoría



}

## Puntos clave {

01

Ingreso de  
datos por  
Formulario  
Web

02

Se visualiza  
gastos  
graficamente

03

Se hace un  
listado de  
gastos  
descargable

04

Optimizacion  
de busquedas

05

Accesibilidad  
en  
multiplataforma

Estructura {

<Sqlit3>

BD



<Streamlit; pd>

Gestor de  
Contenidos



Graficos

<Matploid>

<St.form>

Formulario de  
gastos



<Streamlit>

Front  
End



}

# Funciones Auxilires {

## BASE DE DATOS SQL

```
# Inicializar la base de datos
def inicializar_base_de_datos():
    conexion = sqlite3.connect(ARCHIVO_BD)
    cursor = conexion.cursor()
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS gastos (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
```

Con librerias pandas o SQLit3 se realiza implementacion. Por pandas para tratamientos de base de datos CSV y manipulacion codigo SQL para la version con BD

## BASE DE DATOS CSV

```
def load_expenses():
    """Loads expense data from the CSV file."""
    global expenses_df
    if os.path.exists(CSV_FILE):
        expenses_df = pd.read_csv(CSV_FILE)
        # Ensure the 'Amount' column is numeric after loading
        expenses_df['Amount'] = pd.to_numeric(expenses_df['Amount'], errors='coe
    else:
```

}

# Funciones Intermedias {

## BUSQUEDAS Y FILTROS

```
# Agrupar gastos por periodo (semanal, mensual, anual)
def agrupar_por_periodo(df, periodo='Mensual'):
    df['fecha'] = pd.to_datetime(df['fecha'])
    if periodo == 'Semanal':
        df['periodo'] = df['fecha'].dt.to_period('W').apply(lambda r: r.start_time)
    elif periodo == 'Mensual':
        df['periodo'] = df['fecha'].dt.to_period('M').apply(lambda r: r.start_time)
    elif periodo == 'Anual':
        df['periodo'] = df['fecha'].dt.to_period('Y').apply(lambda r: r.start_time)
    else:
        df['periodo'] = df['fecha'] # Diario
```

Con el apoyo de pandas, streamlit y operaciones de toma de decisiones se realiza las metricas de gastos.

## METRICAS Y ANALISIS TEMPORAL

```
# Tabla y métricas del análisis temporal
st.header("📄 Gastos Filtrados (Análisis Temporal)")
if not gastos_filtrados_df.empty:
    st.dataframe(gastos_filtrados_df[['producto', 'categoria', 'monto', 'fecha']].reset_index())
    total = gastos_filtrados_df['monto'].sum()
    promedio = gastos_filtrados_df['monto'].mean()
    st.metric("💰 Total Gastos (Filtrado)", f"Q{total:,.2f}")
    st.metric("📈 Promedio por Gasto", f"Q{promedio:,.2f}")
else:
```

}



# Funciones FRONTEND {

## STREMLIT FORMS

```
# Formulario para agregar gasto
st.header("🔢 Agregar Nuevo Gasto")
with st.form("Formulario de Gasto"):
    col1, col2 = st.columns(2)
    with col1:
        producto = st.text_input("Producto", placeholder="Ej. Café, Gasolina...")
    with col2:
        categoria = st.selectbox("Categoría", options=CATEGORIAS)

    # Guardar datos en base de datos
```

Con librerías streamlit para  
mostrar el FrontEnd y  
MATPLOTLIB para graficas de  
las proyecciones de gastos en  
el tiempo

## MATPLOTLIB FIGURE

```
# 🍷 Gráfica de pastel: TOTAL DE TODOS LOS TIEMPOS (sin filtros)
st.subheader("🍷 Distribución Total de Gastos por Categoría")
if not todos_los_gastos_df.empty:
    totales_por_categoria = todos_los_gastos_df.groupby('categoria')['monto'].sum()
    fig_pastel, ax_pastel = plt.subplots(figsize=(6, 6))
    ax_pastel.pie(totales_por_categoria, labels=totales_por_categoria.index, autopct='%1.1f%%')
    ax_pastel.set_title("Distribución Global (Todo el historial)")
    ax_pastel.axis('equal')
    st.pyplot(fig_pastel)
else:
    st.info("No hay datos históricos para mostrar.")
```

}

Estructura {

<Sqlit3>

BD



<Streamlit; pd>

Gestor de  
Contenidos



Graficos

<Matploid>

<St.form>

Formulario de  
gastos



<Streamlit>

Front  
End



}

# El Código {

Vamos a visual code a revisar el  
codigo

}

{ Version ya en la nube de streamlit  
<https://gasto5000.streamlit.app/>

}



}

# Planificación {



## FASE 1

Propuesta de proyecto , lluvia de ideas y eleccion del proyecto, un gestor dee gastos con proyecciones.

## FASE 2

Verificacion de que librerias nos ayudan a resolver el proyecto escogido

## FASE 3

Levantamiento de los entornos de programacion, librerias y planeacion de la base de datos, estructuras de cotejo y graficacion

## FASE 4

Programacion en version de base de datos por CSV, se define stremlit para muestra e iteraccion con el usuario

ACTUALIDAD

## FASE 5

Subirlo a un entorno real (Web) y entrega del primer prototipo.

}

PES 2025 Programacion I

Gracias {

<Por="Mario castro", "Diego Cham">

}