```
%pip install wfdb pandas matplotlib seaborn
```

```
Requirement already satisfied: wfdb in c:\users\pevv2\onedrive\documentos\tcd\versionproyecto\datawr
Requirement already satisfied: pandas in c:\users\pevv2\onedrive\documentos\tcd\versionproyecto\data
Requirement already satisfied: matplotlib in c:\users\pevv2\onedrive\documentos\tcd\versionproyecto\
Requirement already satisfied: seaborn in c:\users\pevv2\onedrive\documentos\tcd\versionproyecto\dat
Requirement already satisfied: aiohttp>=3.10.11 in c:\users\pevv2\onedrive\documentos\tcd\versionpro
Requirement already satisfied: fsspec>=2023.10.0 in c:\users\pevv2\onedrive\documentos\tcd\versionpr
Requirement already satisfied: numpy>=1.26.4 in c:\users\pevv2\onedrive\documentos\tcd\versionproyec
Requirement already satisfied: requests>=2.8.1 in c:\users\pevv2\onedrive\documentos\tcd\versionproy
Requirement already satisfied: scipy>=1.13.0 in c:\users\pevv2\onedrive\documentos\tcd\versionproyec
Requirement already satisfied: soundfile>=0.10.0 in c:\users\pevv2\onedrive\documentos\tcd\versionpr
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\pevv2\onedrive\documentos\tcd\vers
Requirement already satisfied: pytz>=2020.1 in c:\users\pevv2\onedrive\documentos\tcd\versionproyect
Requirement already satisfied: tzdata>=2022.7 in c:\users\pevv2\onedrive\documentos\tcd\versionproye
Requirement already satisfied: contourpy>=1.0.1 in c:\users\pevv2\onedrive\documentos\tcd\versionpro
Requirement already satisfied: cycler>=0.10 in c:\users\pevv2\onedrive\documentos\tcd\versionproyect
Requirement already satisfied: fonttools>=4.22.0 in c:\users\pevv2\onedrive\documentos\tcd\versionpr
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\pevv2\onedrive\documentos\tcd\versionpr
Requirement already satisfied: packaging>=20.0 in c:\users\pevv2\onedrive\documentos\tcd\versionproy
Requirement already satisfied: pillow>=8 in c:\users\pevv2\onedrive\documentos\tcd\versionproyecto\d
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\pevv2\onedrive\documentos\tcd\versionpro
Requirement already satisfied: aiohappyeyeballs>=2.5.0 in c:\users\pevv2\onedrive\documentos\tcd\ver
Requirement already satisfied: aiosignal>=1.1.2 in c:\users\pevv2\onedrive\documentos\tcd\versionpro
Requirement already satisfied: attrs>=17.3.0 in c:\users\pevv2\onedrive\documentos\tcd\versionproyec
Requirement already satisfied: frozenlist>=1.1.1 in c:\users\pevv2\onedrive\documentos\tcd\versionpr
Requirement already satisfied: multidict<7.0,>=4.5 in c:\users\pevv2\onedrive\documentos\tcd\version
Requirement already satisfied: propcache>=0.2.0 in c:\users\pevv2\onedrive\documentos\tcd\versionpro
Requirement already satisfied: yarl<2.0,>=1.17.0 in c:\users\pevv2\onedrive\documentos\tcd\versionpr
Requirement already satisfied: six>=1.5 in c:\users\pevv2\onedrive\documentos\tcd\versionproyecto\da
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\pevv2\onedrive\documentos\tcd\ve
Requirement already satisfied: idna<4,>=2.5 in c:\users\pevv2\onedrive\documentos\tcd\versionproyect
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\pevv2\onedrive\documentos\tcd\versionp
Requirement already satisfied: certifi>=2017.4.17 in c:\users\pevv2\onedrive\documentos\tcd\versionp
Requirement already satisfied: cffi>=1.0 in c:\users\pevv2\onedrive\documentos\tcd\versionproyecto\d
Requirement already satisfied: pycparser in c:\users\pevv2\onedrive\documentos\tcd\versionproyecto\d
Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 24.2 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
import wfdb
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns



# Ruta local donde están tus registros
dataset_dir = r'c:\Users\pevv2\OneDrive\Documentos\TCD\versionProyecto\DataWra\mit-bih'

# Listar los registros: .hea
registros = [f.replace('.hea', '') for f in os.listdir(dataset_dir) if f.endswith('.hea')]
print(f"Registros encontrados: {registros}")

# Diccionario de descripciones comunes
ann_label = {
    'N': 'Latido normal',
    'L': 'Latido de bloqueo de rama izquierda',
    'R': 'Latido de bloqueo de rama derecha',
    'V': 'Latido ventricular prematuro',
    'A': 'Latido auricular prematuro',
    'F': 'Latido de fusión',
```

```
        '/': 'Latido de fusión ventricular',
        'f': 'Latido de fusión auricular',
        'j': 'Latido de escape nodal',
        'E': 'Latido de escape ventricular',
        'a': 'Latido auricular aberrante',
        'J': 'Latido de escape de la unión',
        'S': 'Latido de marcapasos',
        'e': 'Latido ventricular aberrante',
        'Q': 'Latido desconocido',
    }

    # Lista para todos los registros
    dfs = []

    for registro in registros:
        print(f"Procesando registro: {registro}")
        try:
            # Leer el registro y anotaciones
            record = wfdb.rdrecord(os.path.join(dataset_dir, registro))
            annotation = wfdb.rdann(os.path.join(dataset_dir, registro), 'atr')

            # Crear DataFrame con las señales
            df_signals = pd.DataFrame(record.p_signal, columns=record.sig_name)
            df_signals['Sample'] = df_signals.index

            # DataFrame de anotaciones
            descripcion = [ann_label.get(s, 'Desconocido') for s in annotation.symbol]
            ann_df = pd.DataFrame({
                'Sample': annotation.sample,
                'Símbolo': annotation.symbol,
                'Descripción': descripcion
            })

            # Merge signals + annotations
            df_merged = pd.merge(df_signals, ann_df, on='Sample', how='left')
            df_merged['Registro'] = registro

            dfs.append(df_merged)

        except Exception as e:
            print(f"Error procesando {registro}: {e}")

    # Dataset final
    df_final = pd.concat(dfs, ignore_index=True)
    print("¡Dataset final creado con éxito!")
```

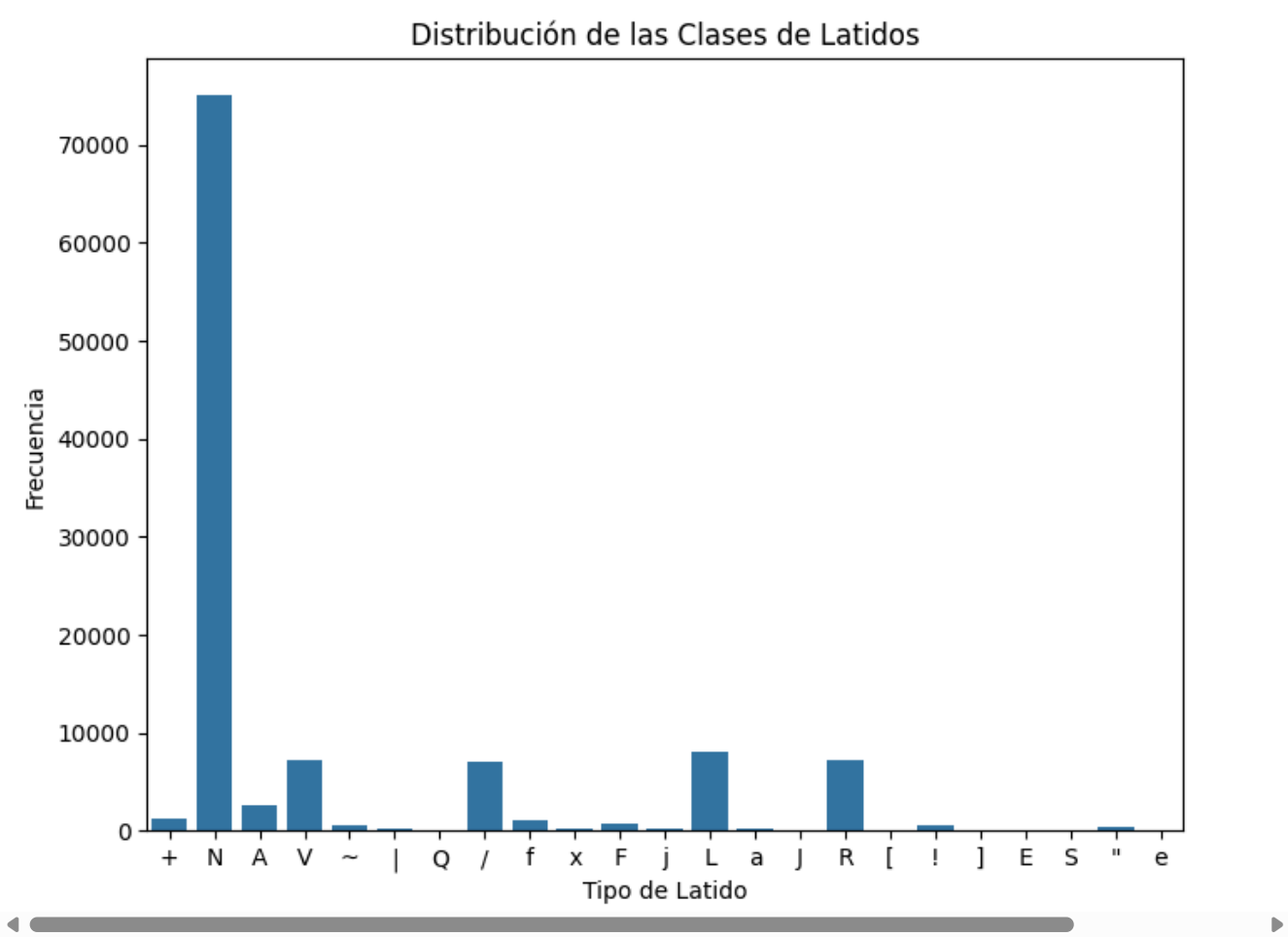
```
Registros encontrados: ['100', '101', '102', '103', '104', '105', '106', '107', '108', '109', '111',
Procesando registro: 100
Procesando registro: 101
Procesando registro: 102
Procesando registro: 103
Procesando registro: 104
Procesando registro: 105
Procesando registro: 106
Procesando registro: 107
Procesando registro: 108
Procesando registro: 109
Procesando registro: 111
Procesando registro: 112
Procesando registro: 113
Procesando registro: 114
Procesando registro: 115
Procesando registro: 116
Procesando registro: 117
```

```
Procesando registro: 118
Procesando registro: 119
Procesando registro: 121
Procesando registro: 122
Procesando registro: 123
Procesando registro: 124
Procesando registro: 200
Procesando registro: 201
Procesando registro: 202
Procesando registro: 203
Procesando registro: 205
Procesando registro: 207
Procesando registro: 208
Procesando registro: 209
Procesando registro: 210
Procesando registro: 212
Procesando registro: 213
Procesando registro: 214
Procesando registro: 215
Procesando registro: 217
Procesando registro: 219
Procesando registro: 220
Procesando registro: 221
Procesando registro: 222
Procesando registro: 223
Procesando registro: 228
Procesando registro: 230
Procesando registro: 231
Procesando registro: 232
Procesando registro: 233
Procesando registro: 234
¡Dataset final creado con éxito!
```

```
# Crear carpeta para guardar las imágenes
output_dir = 'AED_images'
if not os.path.exists(output_dir):
    os.makedirs(output_dir)
```

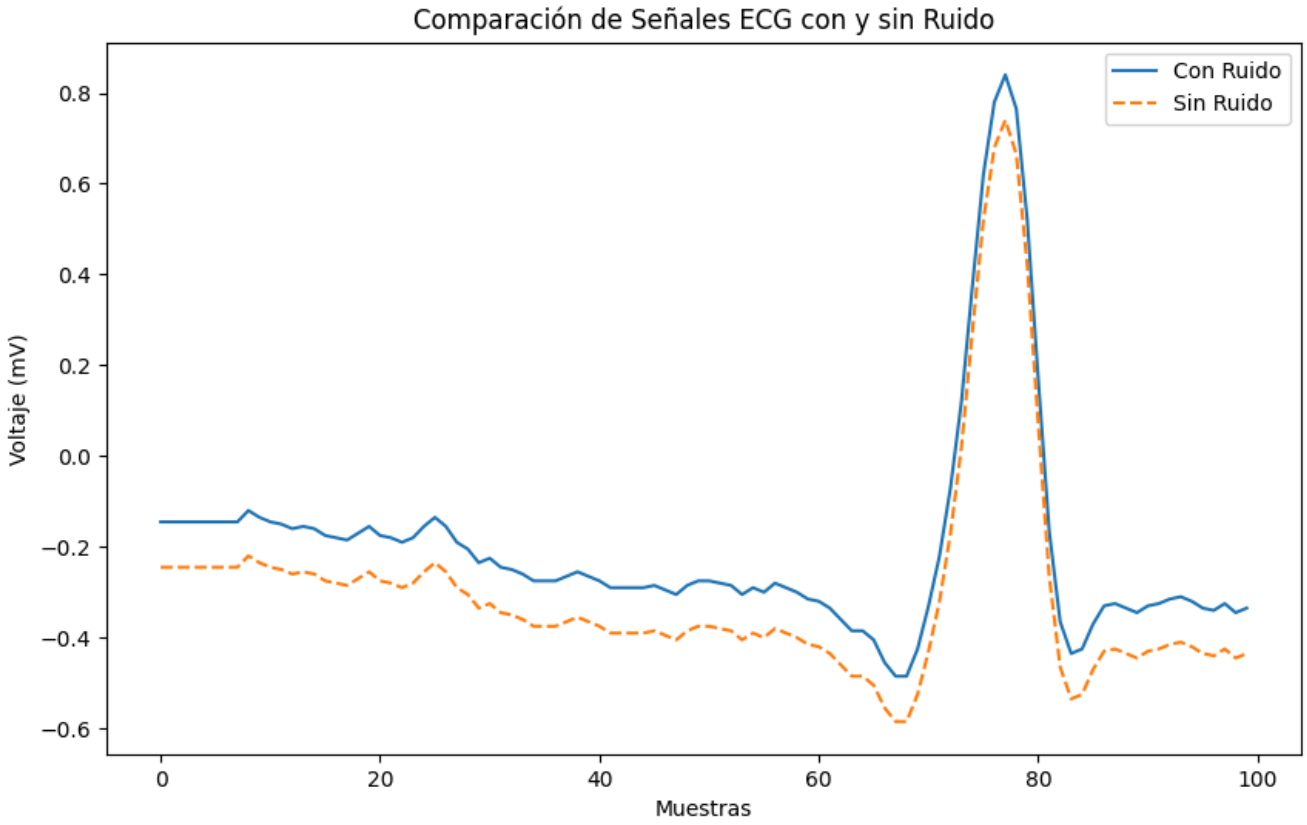
1. Desbalance de clases: Visualizar distribución de clases

```
plt.figure(figsize=(8, 6))
sns.countplot(x='Símbolo', data=df_final)
plt.title('Distribución de las Clases de Latidos')
plt.xlabel('Tipo de Latido')
plt.ylabel('Frecuencia')
plt.savefig(os.path.join(output_dir, 'distribucion_clases.png'))
plt.show()
plt.close()
```

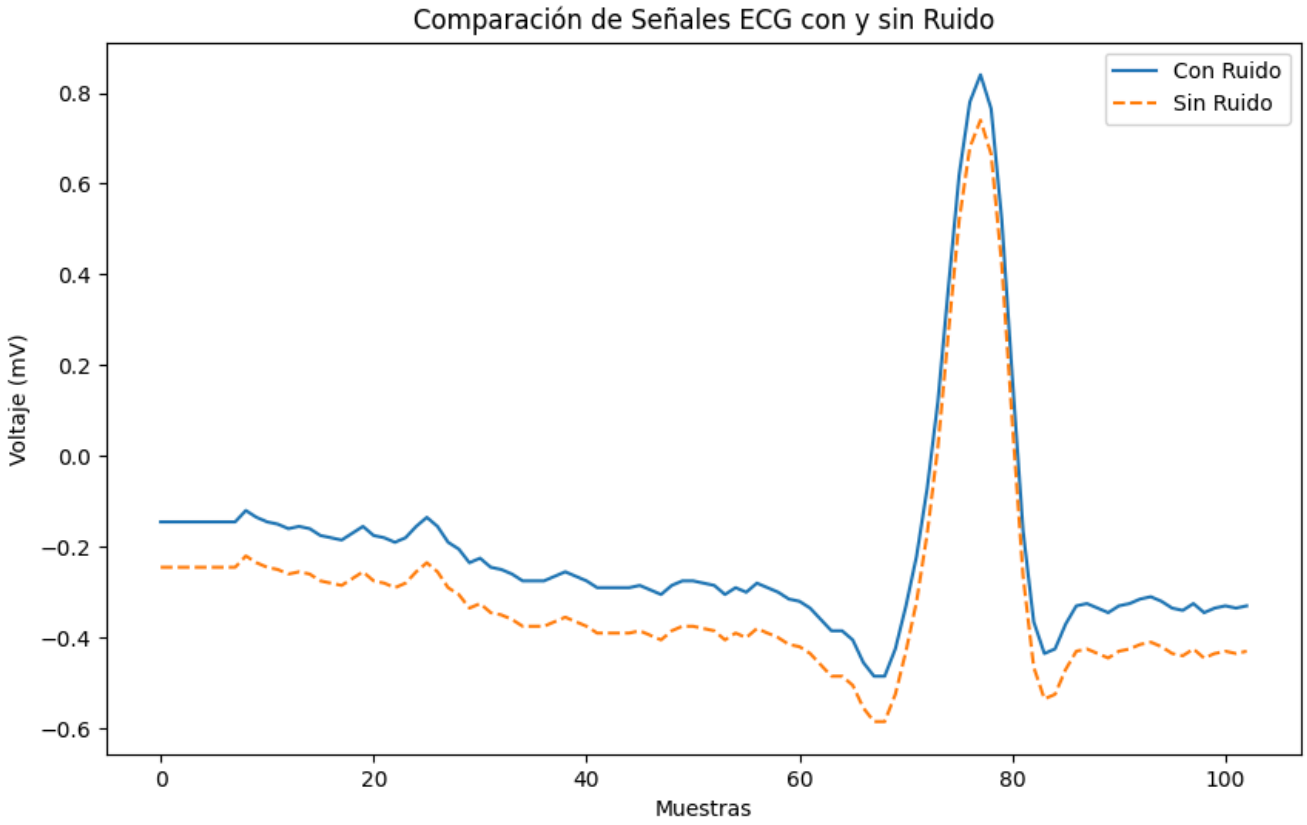


-2. Análisis de ruido y artefactos: Comparación de las señales con y sin ruido Suponiendo que las señales con ruido son las originales y sin ruido son las filtradas (ejemplo hipotético) Para demostrarlo, vamos a comparar las primeras 100 muestras de una señal (por ejemplo, MLII)

```
plt.figure(figsize=(10, 6))
plt.plot(df_final['Sample'][:100], df_final['MLII'][:100], label='Con Ruido')
filtered_signal = df_final['MLII'][:100] - 0.1
plt.plot(df_final['Sample'][:100], filtered_signal, label='Sin Ruido', linestyle='--')
plt.title('Comparación de Señales ECG con y sin Ruido')
plt.xlabel('Muestras')
plt.ylabel('Voltaje (mV)')
plt.legend()
plt.savefig(os.path.join(output_dir, 'senales_contra_sin_ruido.png'))
plt.show()
plt.close()
```

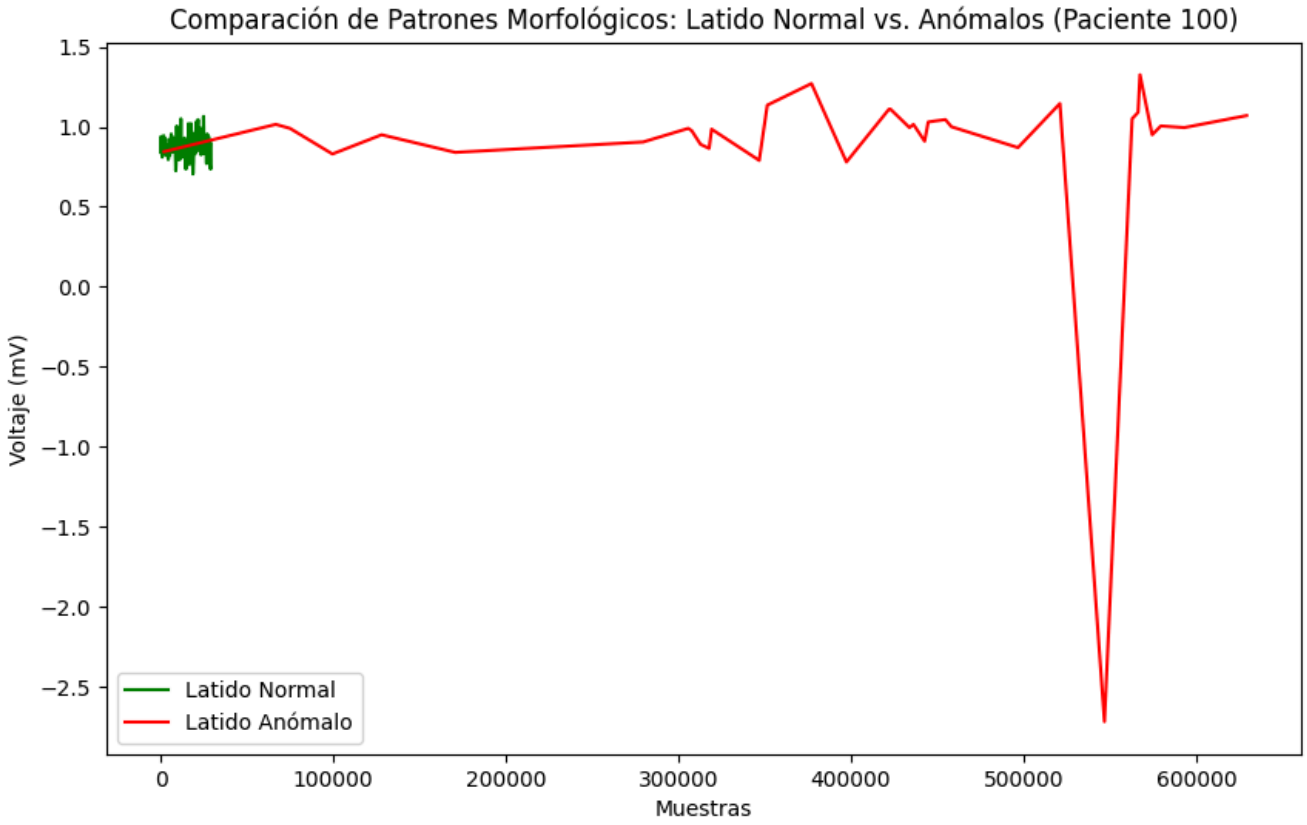



```
plt.figure(figsize=(10, 6))
plt.plot(df_final['Sample'][:103], df_final['MLII'][:103], label='Con Ruido')
filtered_signal = df_final['MLII'][:103] - 0.1
plt.plot(df_final['Sample'][:103], filtered_signal, label='Sin Ruido', linestyle='--')
plt.title('Comparación de Señales ECG con y sin Ruido paciente 103')
plt.xlabel('Muestras')
plt.ylabel('Voltaje (mV)')
plt.legend()
plt.savefig(os.path.join(output_dir, 'senales_contra_sin_ruido.png'))
plt.show()
plt.close()
```



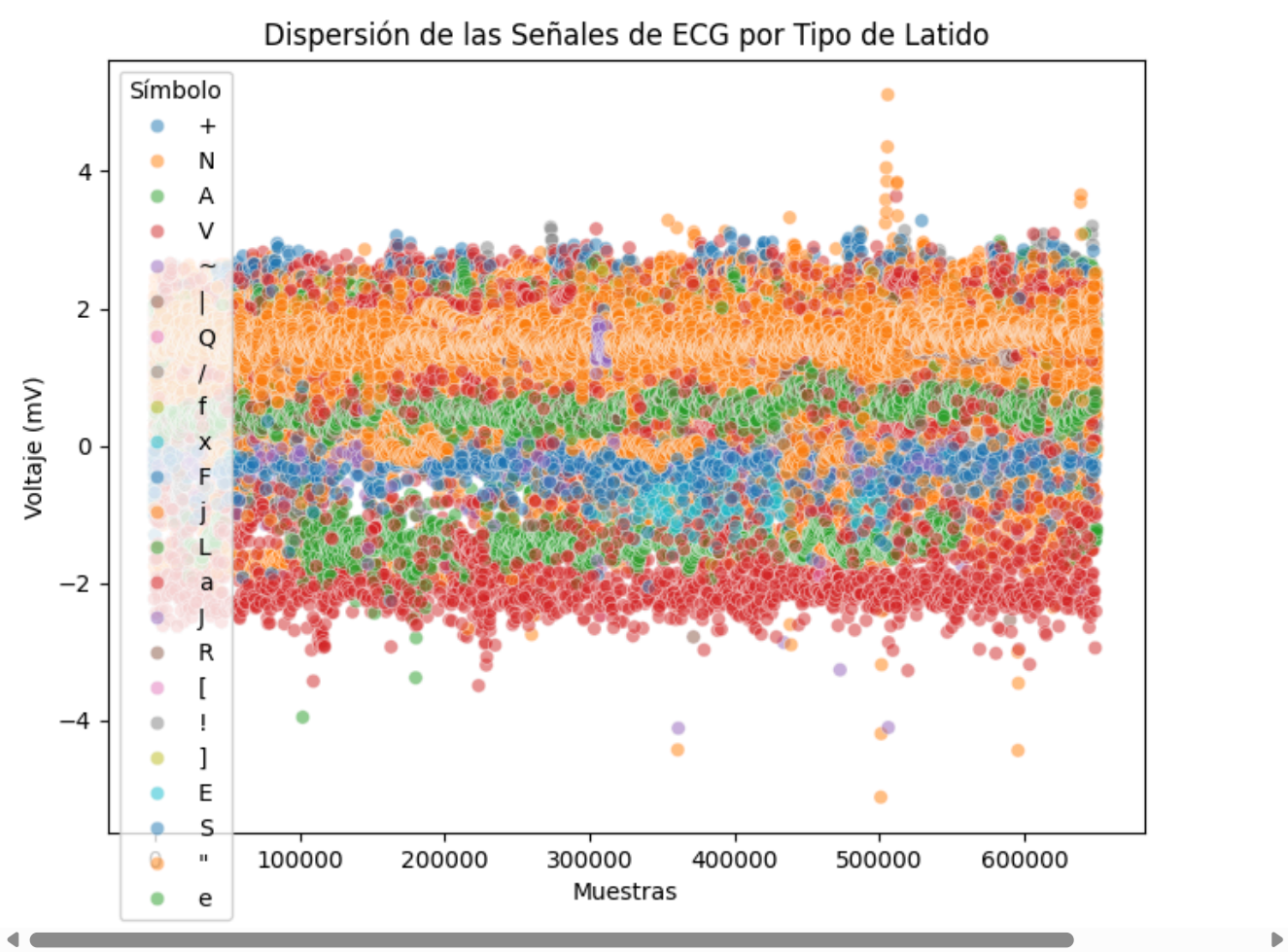
3. Patrones morfológicos: Comparar latidos normales vs. latidos anómalos Seleccionamos una muestra de latidos normales (N) y latidos anómalos (V) para su comparación

```
paciente = '100'
# Símbolos de latidos anómalos (todos excepto 'N')
anomalos = [k for k in ann_label.keys() if k != 'N']

normal_beat = df_final[(df_final['Símbolo'] == 'N') & (df_final['Registro'] == paciente)].iloc[:100]
abnormal_beat = df_final[(df_final['Símbolo'].isin(anomalos)) & (df_final['Registro'] == paciente)].iloc

plt.figure(figsize=(10, 6))
plt.plot(normal_beat['Sample'], normal_beat['MLII'], label='Latido Normal', color='green')
plt.plot(abnormal_beat['Sample'], abnormal_beat['MLII'], label='Latido Anómalo', color='red')
plt.title(f'Comparación de Patrones Morfológicos: Latido Normal vs. Anómalos (Paciente {paciente})')
plt.xlabel('Muestras')
plt.ylabel('Voltaje (mV)')
plt.legend()
plt.savefig(os.path.join(output_dir, f'patrones_morfo_latidos_paciente_{paciente}.png'))
plt.show()
plt.close()
```

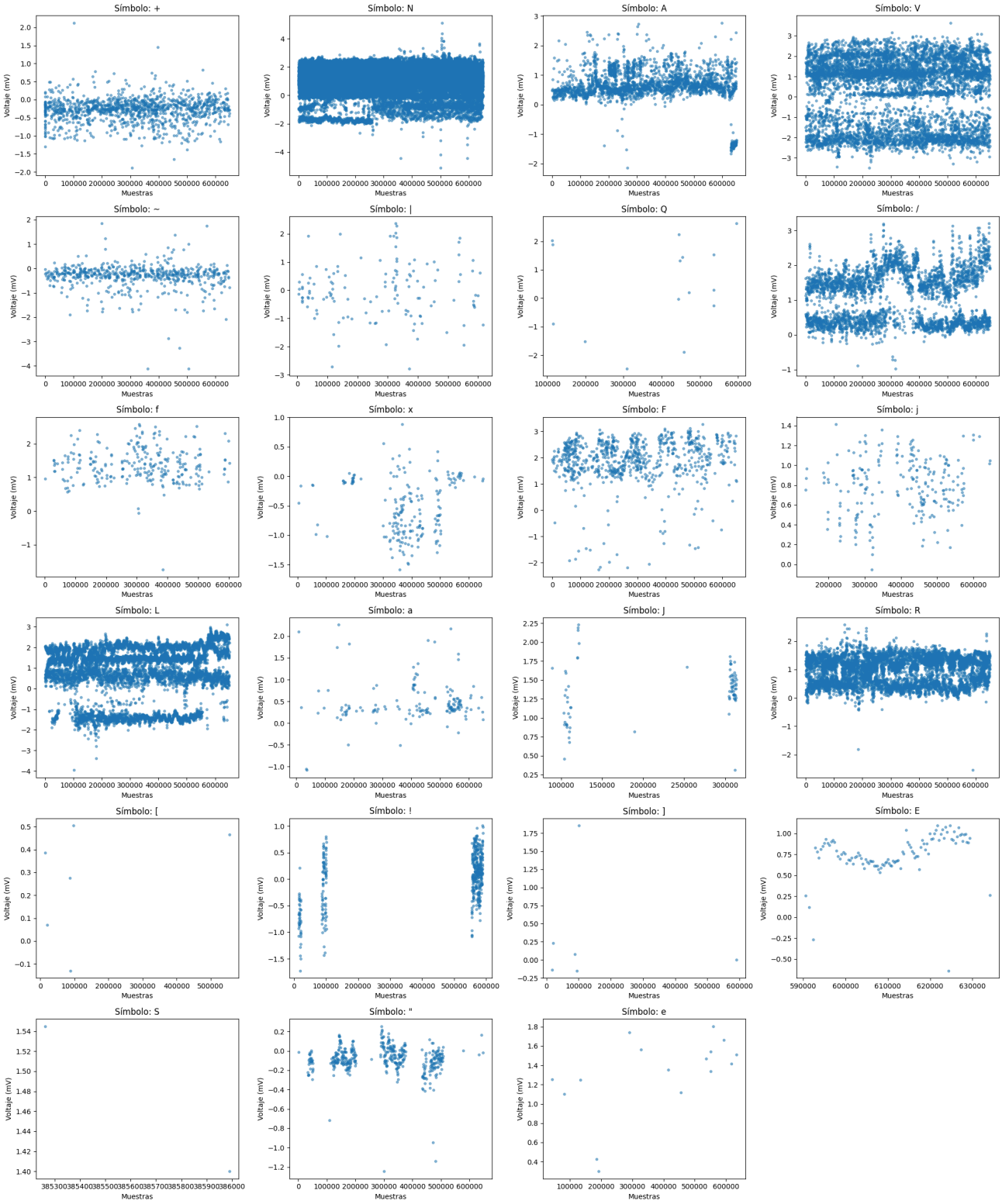


4. Análisis de la dispersión de las señales

```
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Sample', y='MLII', data=df_final, hue='Símbolo', palette='tab10', alpha=0.5)
plt.title('Dispersión de las Señales de ECG por Tipo de Latido')
plt.xlabel('Muestras')
plt.ylabel('Voltaje (mV)')
plt.savefig(os.path.join(output_dir, 'dispercion_senales_ecg.png'))
plt.show()
plt.close()
```



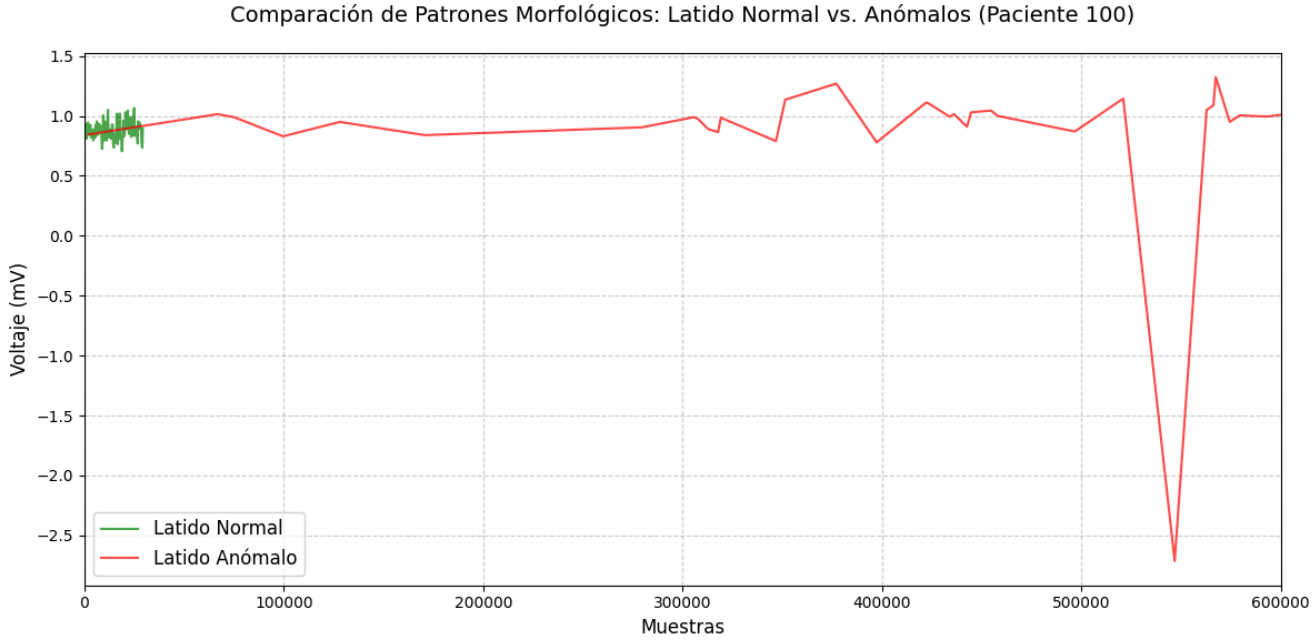

```
simbolos = df_final['Símbolo'].dropna().unique()
n = len(simbolos)
cols = 4
rows = (n + cols - 1) // cols

plt.figure(figsize=(5 * cols, 4 * rows))
for i, simbolo in enumerate(simbolos, 1):
    plt.subplot(rows, cols, i)
    subset = df_final[df_final['Símbolo'] == simbolo]
    plt.scatter(subset['Sample'], subset['MLII'], alpha=0.5, s=10)
    plt.title(f'Símbolo: {simbolo}')
    plt.xlabel('Muestras')
    plt.ylabel('Voltaje (mV)')
    plt.tight_layout()

plt.savefig(os.path.join(output_dir, 'dispercion_senales_por_simbolo.png'))
plt.show()
plt.close()
```




```
import os
import matplotlib.pyplot as plt
import pandas as pd

def comparar_latidos_paciente(df_final, ann_label, paciente='100', output_dir='output'):
    """
    Compara los patrones morfológicos de latidos normales vs. anómalos para un paciente específico
    y guarda el gráfico resultante.

    Parámetros:
```

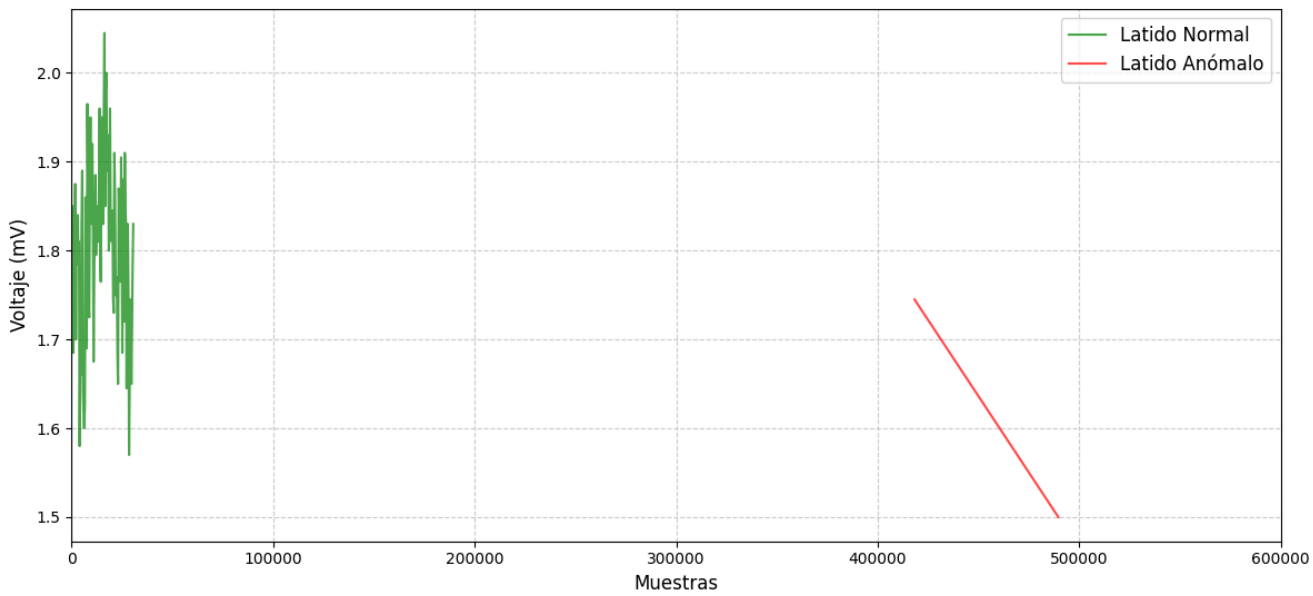
```
            df_final (DataFrame): DataFrame con los datos de los latidos
            ann_label (dict): Diccionario con las anotaciones de los latidos
            paciente (str): ID del paciente a analizar
            output_dir (str): Directorio donde guardar el gráfico
        """

        # Crear directorio de salida si no existe
        os.makedirs(output_dir, exist_ok=True)

        # Símbolos de latidos anómalos (todos excepto 'N')
        anomalos = [k for k in ann_label.keys() if k != 'N']

        # Filtrar latidos normales y anómalos para el paciente
        normal_beat = df_final[(df_final['Símbolo'] == 'N') & (df_final['Registro'] == paciente)].iloc[:100]
        abnormal_beat = df_final[(df_final['Símbolo'].isin(anomalos)) & (df_final['Registro'] == paciente)].

        # Verificar que hay datos suficientes
        if len(normal_beat) == 0:
            print(f"No se encontraron latidos normales para el paciente {paciente}")
            return
        if len(abnormal_beat) == 0:
            print(f"No se encontraron latidos anómalos para el paciente {paciente}")
            return

        print(f"Latidos normales encontrados: {len(normal_beat)}")
        print(f"Latidos anómalos encontrados: {len(abnormal_beat)}")

        # Configurar el gráfico
        plt.figure(figsize=(12, 6))

        # Graficar latidos normales (verde) y anómalos (rojo)
        plt.plot(normal_beat['Sample'], normal_beat['MLII'],
                 label='Latido Normal', color='green', alpha=0.7)
        plt.plot(abnormal_beat['Sample'], abnormal_beat['MLII'],
                 label='Latido Anómalo', color='red', alpha=0.7)

        # Configurar título y etiquetas
        plt.title(f'Comparación de Patrones Morfológicos: Latido Normal vs. Anómalos (Paciente {paciente})',
                  fontsize=14, pad=20)
        plt.xlabel('Muestras', fontsize=12)
        plt.ylabel('Voltaje (mV)', fontsize=12)

        # Configurar ejes y cuadrícula
        plt.xlim(0, 600000)
        plt.grid(True, linestyle='--', alpha=0.6)
        plt.legend(fontsize=12)

        # Ajustar diseño
        plt.tight_layout()

        # Guardar y mostrar
        output_path = os.path.join(output_dir, f'patrones_morfo_latidos_paciente_{paciente}.png')
        plt.savefig(output_path, dpi=300, bbox_inches='tight')
        print(f"Gráfico guardado en: {output_path}")
        plt.show()
        plt.close()

    # Ejemplo de uso:
    comparar_latidos_paciente(df_final, ann_label, paciente='100')
```

```
Latidos normales encontrados: 100
Latidos anómalos encontrados: 34
Gráfico guardado en: output\patrones_morfo_latidos_paciente_100.png
```



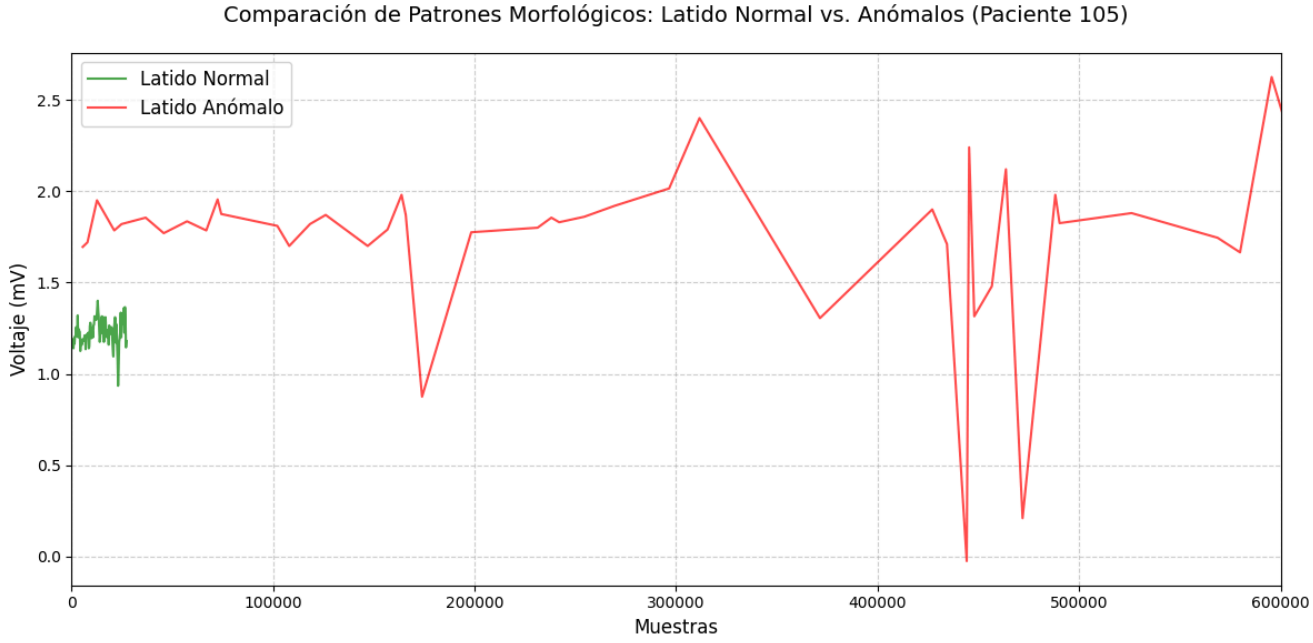

```
comparar_latidos_paciente(df_final, ann_label, paciente='103')
```

```
Latidos normales encontrados: 100
Latidos anómalos encontrados: 2
Gráfico guardado en: output\patrones_morfo_latidos_paciente_103.png
```



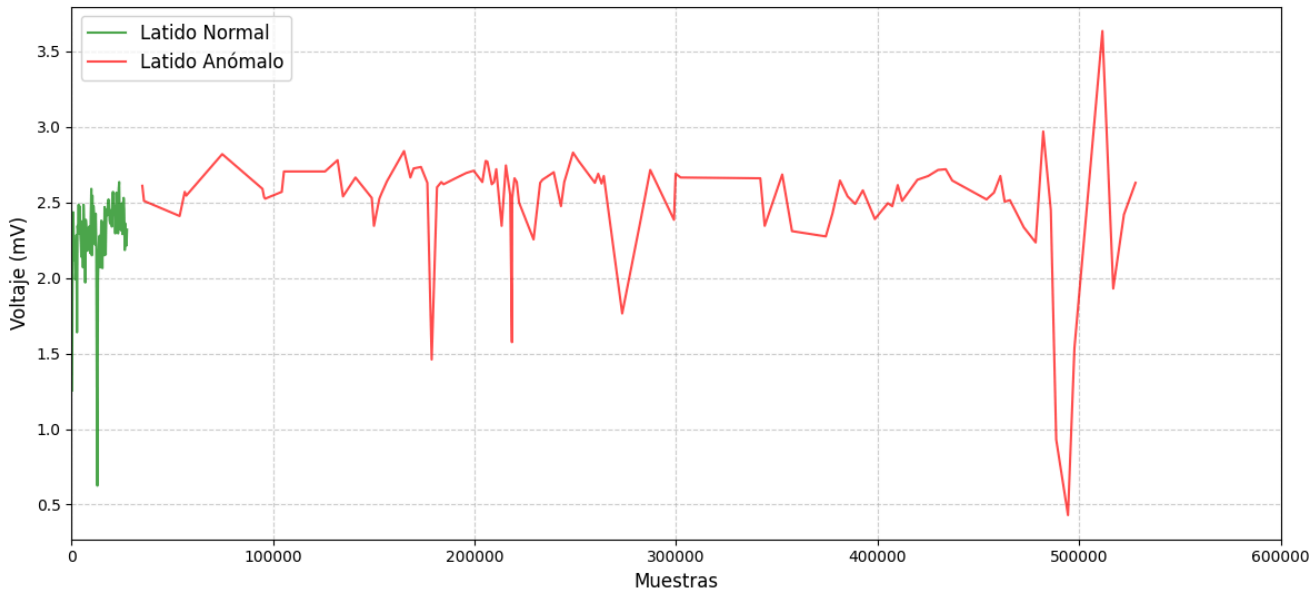

```
comparar_latidos_paciente(df_final, ann_label, paciente='105')
```

```
Latidos normales encontrados: 100
Latidos anómalos encontrados: 46
Gráfico guardado en: output\patrones_morfo_latidos_paciente_105.png
```




```
comparar_latidos_paciente(df_final, ann_label, paciente='116')
```

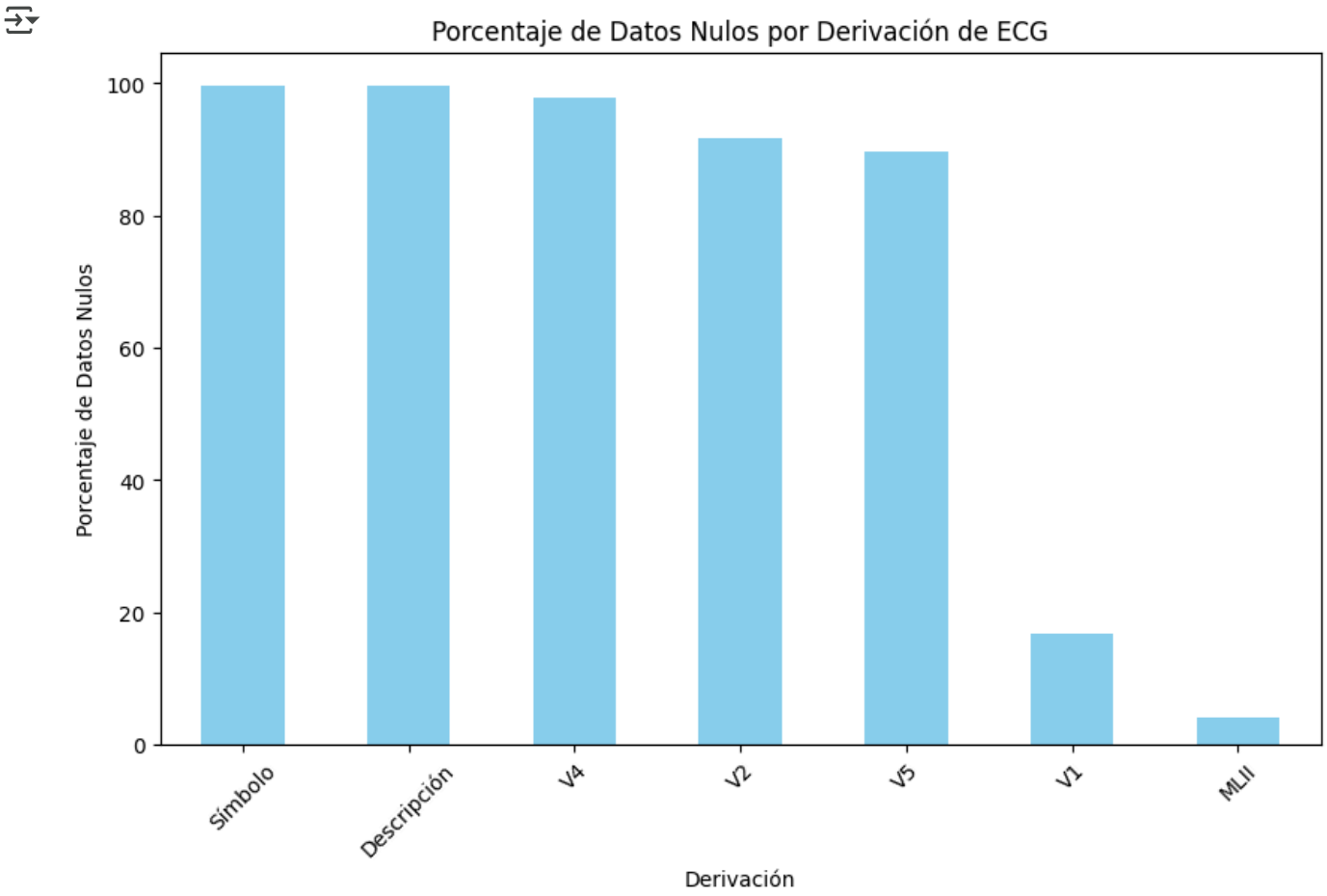
```
Latidos normales encontrados: 100
Latidos anómalos encontrados: 100
Gráfico guardado en: output\patrones_morfo_latidos_paciente_116.png
```



5. Análisis de la cantidad de datos nulos en las distintas derivaciones

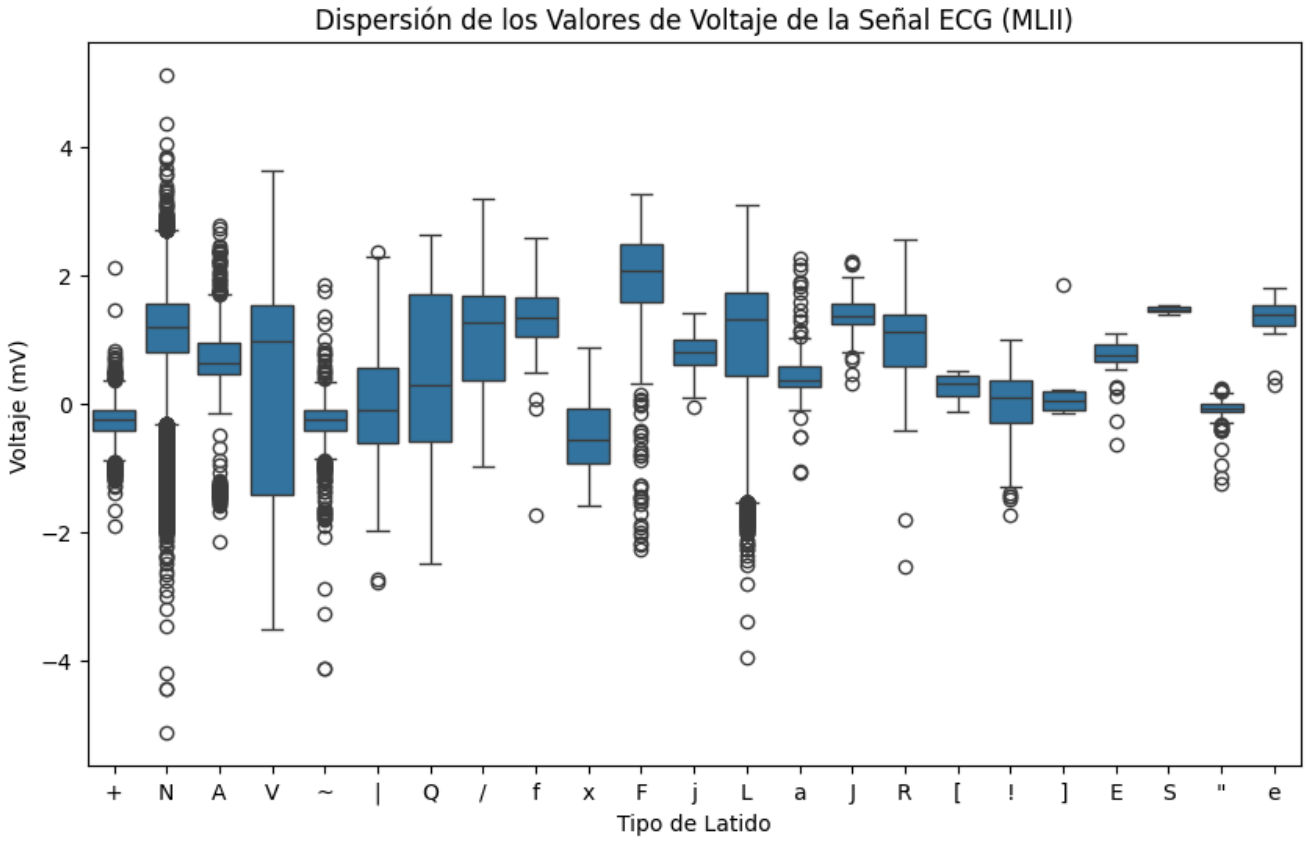
```
plt.figure(figsize=(10, 6))
missing_data = df_final.isnull().sum() / len(df_final) * 100  # Porcentaje de valores nulos
missing_data = missing_data[missing_data > 0]  # Solo mostrar derivaciones con valores nulos
missing_data.sort_values(ascending=False).plot(kind='bar', color='skyblue')
plt.title('Porcentaje de Datos Nulos por Derivación de ECG')
plt.xlabel('Derivación')
```

```
plt.ylabel('Porcentaje de Datos Nulos')
plt.xticks(rotation=45)
plt.savefig(os.path.join(output_dir, 'porcentaje_datos_nulos.png'))
plt.show()
plt.close()
```

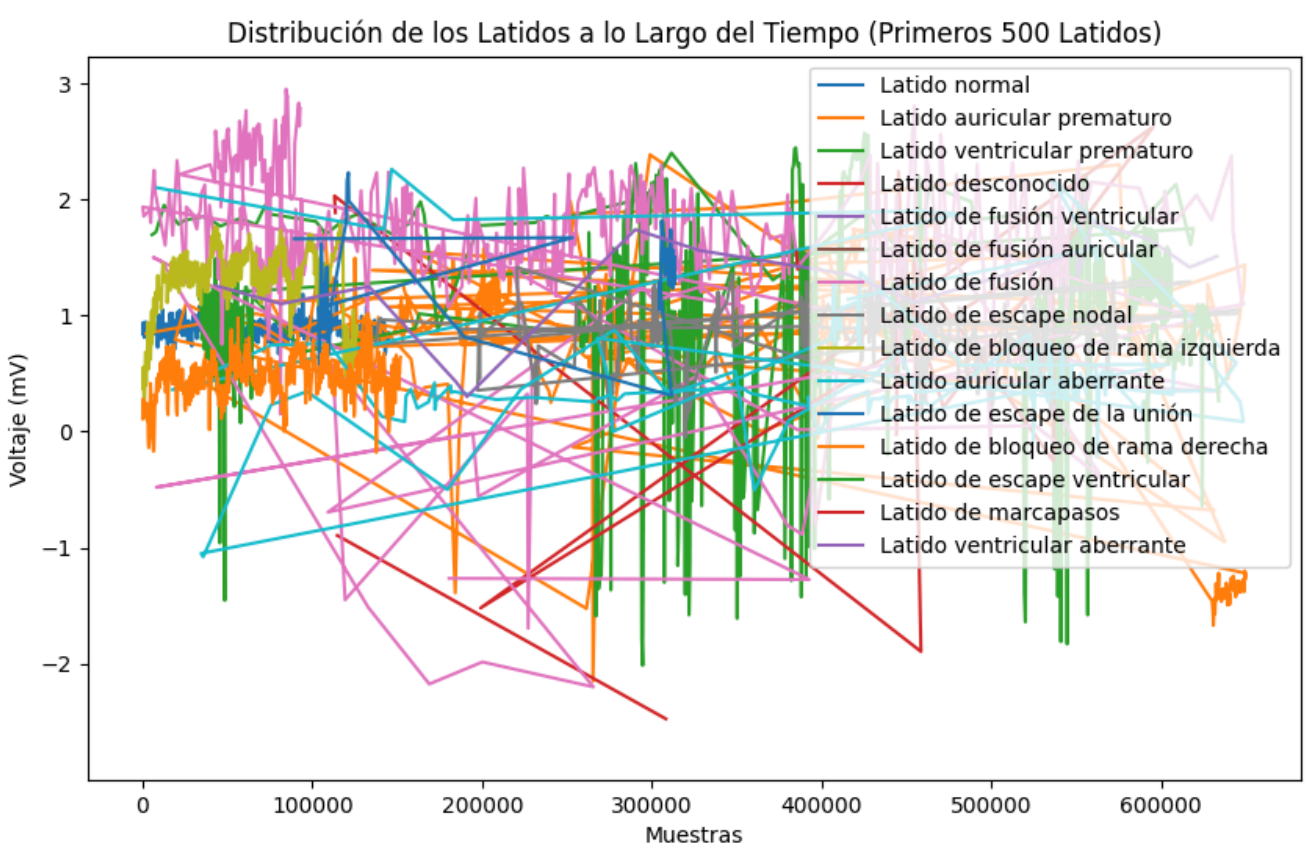


6. Análisis de la dispersión de los valores de voltaje de la señal ECG (detectar outliers)

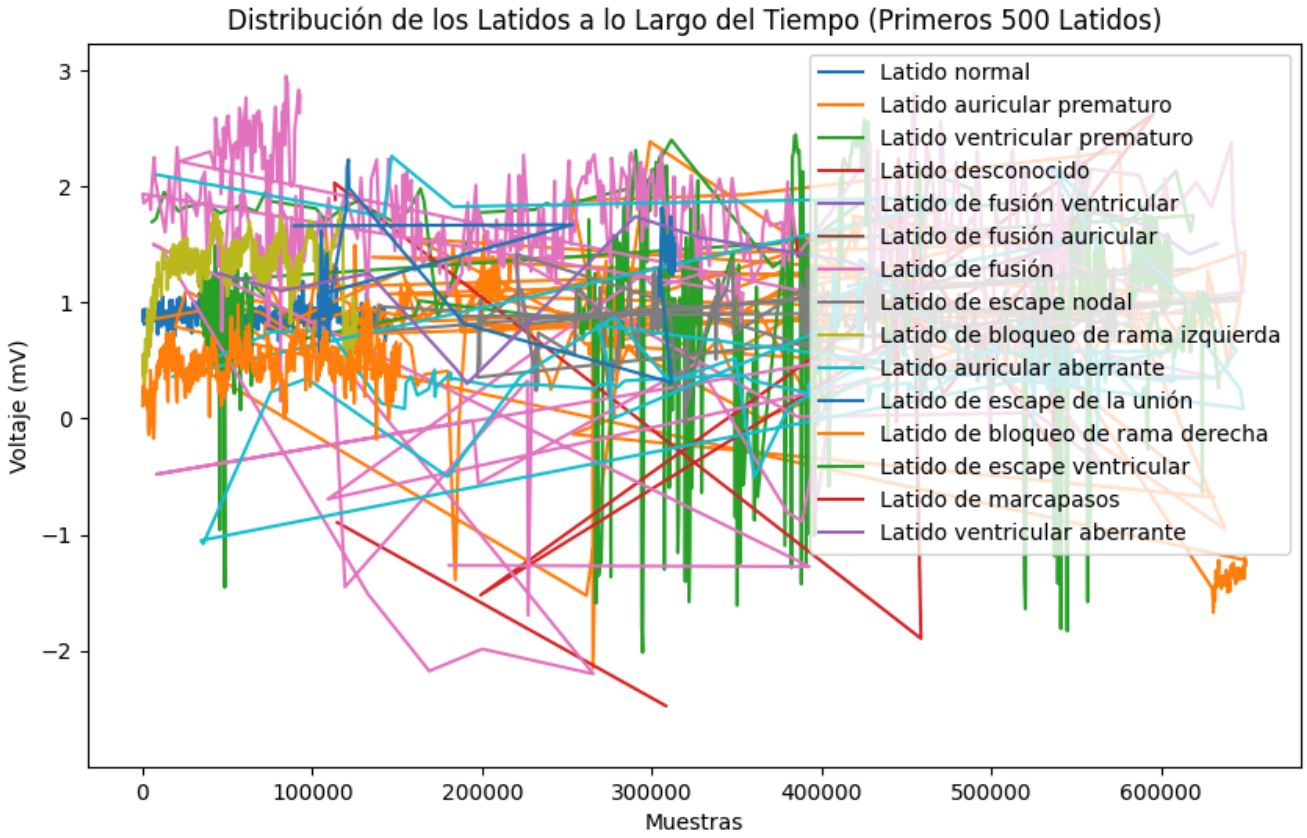
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Símbolo', y='MLII', data=df_final)
plt.title('Dispersión de los Valores de Voltaje de la Señal ECG (MLII)')
plt.xlabel('Tipo de Latido')
plt.ylabel('Voltaje (mV)')
plt.savefig(os.path.join(output_dir, 'dispercion_valores_ecg.png'))
plt.show()
plt.close()
```



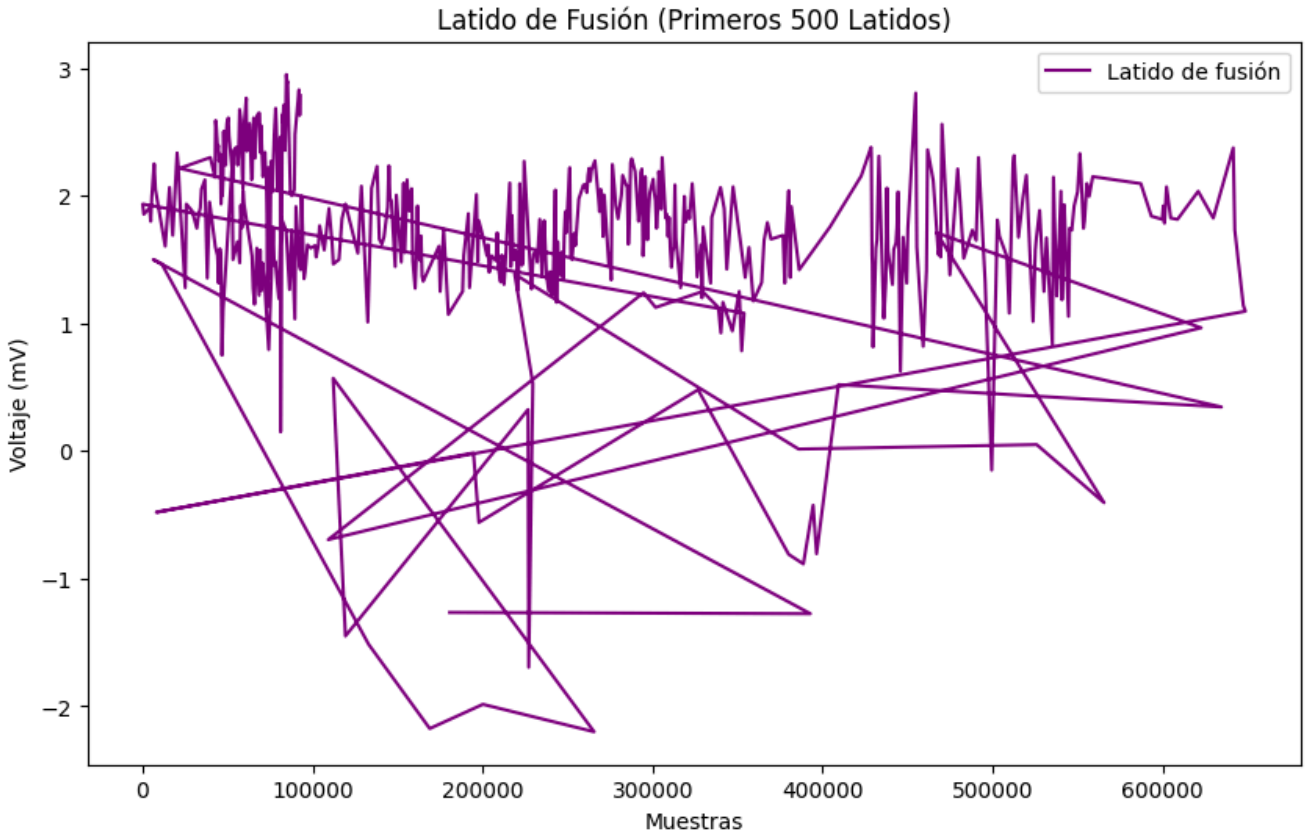

```
plt.figure(figsize=(10, 6))
for label in df_final['Símbolo'].unique():
    desc = ann_label.get(label, "Desconocido")
    if desc == "Desconocido":
        continue
    subset = df_final[df_final['Símbolo'] == label]
    plt.plot(subset['Sample'][:500], subset['MLII'][:500], label=desc)
plt.title('Distribución de los Latidos a lo Largo del Tiempo (Primeros 500 Latidos)')
plt.xlabel('Muestras')
plt.ylabel('Voltaje (mV)')
plt.legend(loc='upper right')
plt.savefig(os.path.join(output_dir, 'distribucion_latidos_tiempo.png'))
plt.show()
plt.close()
```



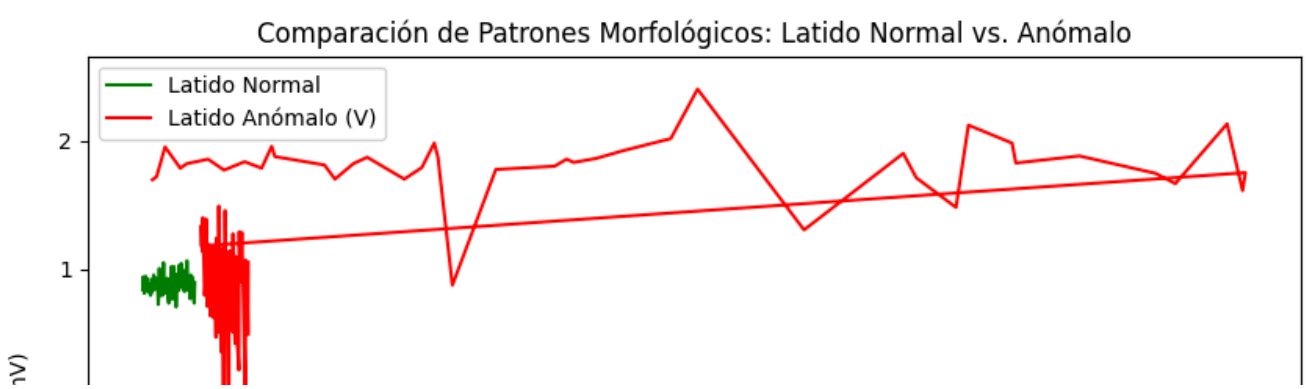

```
plt.figure(figsize=(10, 6))
for label in df_final['Símbolo'].unique():
    desc = ann_label.get(label, "Desconocido")
    if desc == "Desconocido":
        continue
    subset = df_final[df_final['Símbolo'] == label]
    plt.plot(subset['Sample'][:500], subset['MLII'][:500], label=desc)
plt.title('Distribución de los Latidos a lo Largo del Tiempo (Primeros 500 Latidos)')
plt.xlabel('Muestras')
plt.ylabel('Voltaje (mV)')
plt.legend(loc='upper right')
plt.savefig(os.path.join(output_dir, 'distribucion_latidos_tiempo.png'))
plt.show()
plt.close()
```




```
# Graficar solo los latidos de fusión ('F')
fusion_label = 'F'
fusion_desc = ann_label.get(fusion_label, "Desconocido")
fusion_subset = df_final[df_final['Símbolo'] == fusion_label]
if not fusion_subset.empty:
    plt.figure(figsize=(10, 6))
    plt.plot(fusion_subset['Sample'][:500], fusion_subset['MLII'][:500], label=fusion_desc, color='purpl
    plt.title('Latido de Fusión (Primeros 500 Latidos)')
    plt.xlabel('Muestras')
    plt.ylabel('Voltaje (mV)')
    plt.legend()
    plt.show()
else:
    print("No se encontraron latidos de fusión ('F') en el dataset.")
```




```
# 8. Comparación de características morfológicas de los latidos normales vs. latidos anómalos
# Seleccionamos una muestra de latidos normales (N) y latidos anómalos (V) para su comparación
normal_beat = df_final[df_final['Símbolo'] == 'N'].iloc[0:100]  # Primeros 100 latidos normales
abnormal_beat = df_final[df_final['Símbolo'] == 'V'].iloc[0:100]  # Primeros 100 latidos ventriculares pre

plt.figure(figsize=(10, 6))
plt.plot(normal_beat['Sample'], normal_beat['MLII'], label='Latido Normal', color='green')
plt.plot(abnormal_beat['Sample'], abnormal_beat['MLII'], label='Latido Anómalo (V)', color='red')
plt.title('Comparación de Patrones Morfológicos: Latido Normal vs. Anómalo')
plt.xlabel('Muestras')
plt.ylabel('Voltaje (mV)')
plt.legend()
plt.savefig(os.path.join(output_dir, 'comparacion_morfo_latidos.png'))
plt.show()
plt.close()
```




```
# 8. Comparación de características morfológicas de los latidos normales vs. latidos anómalos
# Seleccionamos una muestra de latidos normales (N) y latidos anómalos (V) para su comparación
normal_beat = df_final[df_final['Símbolo'] == 'N'].iloc[0:1000]  # Primeros 100 latidos normales
abnormal_beat = df_final[df_final['Símbolo'] == 'V'].iloc[0:1000]  # Primeros 100 latidos ventriculares pr

plt.figure(figsize=(10, 6))
plt.plot(normal_beat['Sample'], normal_beat['MLII'], label='Latido Normal', color='green')
plt.plot(abnormal_beat['Sample'], abnormal_beat['MLII'], label='Latido Anómalo (V)', color='red')
plt.title('Comparación de Patrones Morfológicos: Latido Normal vs. Anómalo')
plt.xlabel('Muestras')
plt.ylabel('Voltaje (mV)')
plt.legend()
plt.savefig(os.path.join(output_dir, 'comparacion_morfo_latidos.png'))
plt.show()
plt.close()
```