In [54]:

```python
import os
import numpy as np
import wfdb
from datetime import datetime
from scipy.signal import butter, sosfiltfilt, iirnotch, filtfilt, find_peaks

# === 1) CONFIGURAÇÕES ===
base_dir         = r'C:\Users\Stella\Desktop\TCC'
arquivo_txt      = os.path.join(base_dir, 'ecg_bruto.txt')
record_name      = 'meu_ecg_filtrado'
full_path        = os.path.join(base_dir, record_name)  # sem extensão
fs               = 977       # Hz
adc_bits         = 12
adc_max          = 2**adc_bits - 1
v_ref            = 3.3       # volts
adc_gain_wfdb    = 1000      # 1 unit WFDB = 1 µV
lowcut, highcut  = 0.5, 45.0 # Hz
notch_freq, Q    = 60.0, 30
min_rr_interval  = 0.3       # s
peak_height      = 0.1       # mV
peak_prominence  = 0.05      # mV

# === 2) LEITURA DO TXT ===
valores = []
with open(arquivo_txt, 'r') as f:
    for linha in f:
        try:
            valores.append(int(linha.strip()))
        except ValueError:
            pass

valores = np.array(valores)
print(f"🔎 {len(valores)} amostras carregadas de '{arquivo_txt}'")

# === 3) CONVERSÃO ADC → mV + OFFSET DC ===
sinal_mv = (valores / adc_max) * v_ref * 1000
sinal_mv -= np.mean(sinal_mv)
print(f"📊 Sinal convertido: min={sinal_mv.min():.2f} mV, max={sinal_mv.max():.2f} mV")

# === 4) PASSA-FAIXA ===
def passa_faixa(x, fs, low, high, ordem=4):
    nyq = fs/2
    sos = butter(ordem, [low/nyq, high/nyq], btype='band', output='sos')
    return sosfiltfilt(sos, x)

sinal_band = passa_faixa(sinal_mv, fs, lowcut, highcut)

# === 5) NOTCH 60 Hz ===
def notch(x, fs, f0, Q):
    b, a = iirnotch(f0, Q, fs)
    return filtfilt(b, a, x)

sinal_clean = notch(sinal_band, fs, notch_freq, Q)

# === 6) NORMALIZAÇÃO (±2 mV) ===
sinal_norm = sinal_clean / np.max(np.abs(sinal_clean)) * 2.0

# === 7) SALVA EM WFDB (.dat + .hea) NO DIRETÓRIO base_dir ===
print("→ Salvando WFDB em:", os.path.join(base_dir, record_name + ".dat"))
```

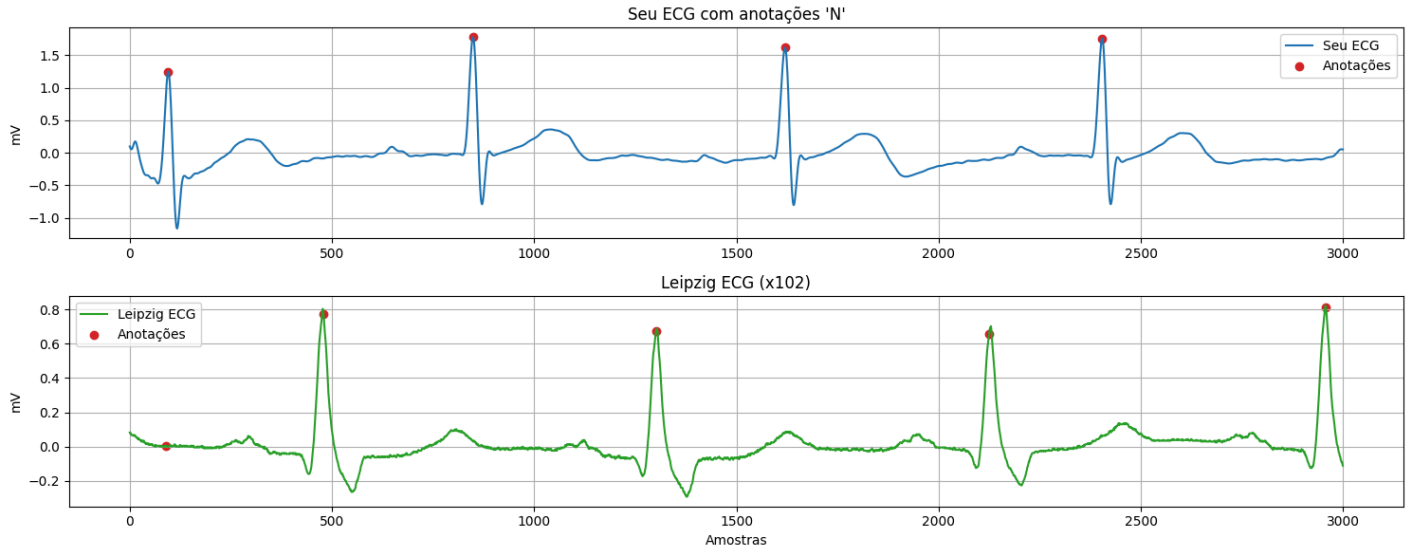
```
wfdb.wrsamp(
    record_name=record_name,   # apenas o nome simples
    write_dir=base_dir,        # aqui vai o caminho da pasta
    fs=fs,
    sig_name=['ECG'],
    units=['mV'],
    p_signal=sinal_norm.reshape(-1, 1),
    fmt=['16'],
    adc_gain=[adc_gain_wfdb],
    baseline=[0],
    comments=[f"Processado: passa-faixa, notch, normalizado em {datetime.now()}"]
)
print("→ Conteúdo da pasta após wrsamp():", os.listdir(base_dir))

# === 8) GERA .atr NO MESMO DIRETÓRIO ===
if peaks.size:
    wfdb.wrann(
        record_name=record_name,   # novamente só o nome
        write_dir=base_dir,        # mesmo diretório
        extension='atr',
        sample=peaks,
        symbol=['N'] * len(peaks),
        fs=fs
    )
    print("→ Arquivos .atr na pasta:",
          [f for f in os.listdir(base_dir) if f.endswith('.atr')])
else:
    print("⚠️ Nenhum pico detectado; .atr não gerado.")
```

```
🔎 14655 amostras carregadas de 'C:\Users\Stella\Desktop\TCC\ecg_bruto.txt'
📊 Sinal convertido: min=-1252.99 mV, max=1848.77 mV
→ Salvando WFDB em: C:\Users\Stella\Desktop\TCC\meu_ecg_filtrado.dat
→ Conteúdo da pasta após wrsamp(): ['.ipynb_checkpoints', 'arduino', 'arduino-ide_nightly-2025041
1_Windows_64bit.zip', 'arduino_para_porta_serial', 'dabatase exp.ipynb', 'database.txt', 'db', 'e
cg_bruto.txt', 'leipzig-heart-center-ecg-database-arrhythmias-in-children-and-patients-with-conge
nital-heart-disease-1.0.0.zip', 'meu_ecg_convertido.dat', 'meu_ecg_convertido.hea', 'meu_ecg_filt
rado.atr', 'meu_ecg_filtrado.dat', 'meu_ecg_filtrado.hea', 'Minha-Amostragem - Copy.ipynb', 'Minh
a-Amostragem.ipynb', 'Minha-Amostragem_v1.pdf', 'minha_amostragem', 'Untitled.ipynb']
→ Arquivos .atr na pasta: ['meu_ecg_filtrado.atr']
```

In [60]:

```
import os
import numpy as np
import matplotlib.pyplot as plt
import wfdb

# === 1) CONFIGURAÇÃO DE CAMINHOS ===
base_dir            = r'C:\Users\Stella\Desktop\TCC'
user_record_path    = os.path.join(base_dir, 'meu_ecg_filtrado')
leipzig_record_path = os.path.join(base_dir, 'db', 'x102')

# === 2) LEITURA DOS REGISTROS WFDB ===
user_rec     = wfdb.rdrecord(user_record_path)
user_ann     = wfdb.rdann(user_record_path, 'atr')
leipzig_rec  = wfdb.rdrecord(leipzig_record_path)
leipzig_ann  = wfdb.rdann(leipzig_record_path, 'atr')

# === 3) PARÂMETROS DE PLOT ===
samples_to_plot = 3000  # cerca de 3 segundos (ajuste se necessário)

# === 4) PLOTAGEM COM SUBPLOTS ===
fig, axes = plt.subplots(2, 1, figsize=(15, 6))
```

```
# ── Seu ECG ──
axes[0].plot(
    user_rec.p_signal[:samples_to_plot, 0],
    label='Seu ECG', color='tab:blue'
)
user_peaks = [s for s in user_ann.sample if s < samples_to_plot]
axes[0].scatter(
    user_peaks,
    user_rec.p_signal[user_peaks, 0],
    color='tab:red', label='Anotações'
)
axes[0].set(
    title="Seu ECG com anotações 'N'",
    ylabel="mV"
)
axes[0].legend()
axes[0].grid(True)

# ── Leipzig ECG ──
axes[1].plot(
    leipzig_rec.p_signal[:samples_to_plot, 0],
    label='Leipzig ECG', color='tab:green'
)
leipzig_peaks = [s for s in leipzig_ann.sample if s < samples_to_plot]
axes[1].scatter(
    leipzig_peaks,
    leipzig_rec.p_signal[leipzig_peaks, 0],
    color='tab:red', label='Anotações'
)
axes[1].set(
    title="Leipzig ECG (x102)",
    xlabel="Amostras",
    ylabel="mV"
)
axes[1].legend()
axes[1].grid(True)

plt.tight_layout()
plt.show()
```



In [ ]:

In [ ]: