```python
import numpy as np

def generate_matrix(Ej: float, r: float, Ec: float, k: float, N: int) -> np.ndarray:
    """
    Generate the matrix representation of the Hamiltonian.

    Parameters:
    - Ej (float): Josephson energy.
    - r (float): Reflectivity.
    - Ec (float): Charging energy.
    - k (float): Wave number.
    - N (int): Size parameter for the matrix. The actual size will be 2*(N+1) x 2*(N+1).

    Returns:
    - np.ndarray: Generated matrix.
    """

    # Initialize the matrix with zeros. The matrix has dimensions [2*(N+1), 2*(N+1)]
    matrix = np.zeros((2*(N+1), 2*(N+1)))

    # Populate the main diagonal and the sub-diagonals.
    for i, idx in enumerate(np.arange(-N, N + 2)):
        # Populate the main diagonal with 4 * Ec * (k + i/2)^2
        k_val = k - (idx // 2) / 2  # Calculate k + i/2 for each pair of indices
        matrix[i, i] = 4 * Ec * k_val ** 2

        # Populate the sub-diagonals and super-diagonals.
        if i + 1 < 2 * (N + 1):
            matrix[i, i + 1] = 0 if i % 2 == 0 else -r * Ej / 2
            matrix[i + 1, i] = 0 if i % 2 == 0 else -r * Ej / 2

        if i + 2 < 2 * (N + 1):
            matrix[i, i + 2] = Ej / 2 if i % 2 == 0 else -Ej / 2
            matrix[i + 2, i] = Ej / 2 if i % 2 == 0 else -Ej / 2

        if i + 3 < 2 * (N + 1):
            matrix[i, i + 3] = - r * Ej / 2 if i % 2 == 0 else 0
            matrix[i + 3, i] = r * Ej / 2 if i % 2 == 0 else 0

    return matrix
```