In [1]:
```
import os
os.environ["PROJ_LIB"] = "C:\\Utilities\\Python\\Anaconda\\Library\\share";
import pandas as pd
import networkx as nx
import community
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap as Basemap
```

In [2]:
```
# read data
airports = pd.read_csv('dane1.csv')
conections = pd.read_csv('polaczenia1.csv')
```

In [3]:
```
# show airports data
airports
```

Out[3]:

	Imię	Pierwsza litera nazwiska
0	Anna	N
1	Katarzyna	K
2	Piotr	P
3	Paweł	P
4	Michał	M
...	...	...
107	Klaudia	K
108	Mateusz	M
109	Julia	J
110	Piotr	P
111	Olga	O

112 rows × 2 columns

In [4]:
```
# show conections info
conections
```

Out[4]:

	column1	Column2	column2	Column4	Imien1	Imien2
0	Anna	N	Katarzyna	L	Anna N	Katarzyna L
1	Anna	N	Piotr	K	Anna N	Piotr K
2	Anna	N	Paweł	W	Anna N	Paweł W
3	Anna	N	Michał	S	Anna N	Michał S
4	Anna	N	Jan	K	Anna N	Jan K
5	Katarzyna	K	Katarzyna	W	Katarzyna K	Katarzyna W
6	Katarzyna	K	Katarzyna	L	Katarzyna K	Katarzyna L
7	Katarzyna	K	Katarzyna	K	Katarzyna K	Katarzyna K
8	Katarzyna	K	Katarzyna	S	Katarzyna K	Katarzyna S
9	Katarzyna	K	Katarzyna	K	Katarzyna K	Katarzyna K
10	Piotr	P	Lewandowski	W	Piotr P	Lewandowski W
11	Piotr	P	Kowalski	W	Piotr P	Kowalski W
12	Piotr	P	Kamiński	W	Piotr P	Kamiński W
13	Piotr	P	Szymański	S	Piotr P	Szymański S
14	Piotr	P	Wójcik	K	Piotr P	Wójcik K
15	Paweł	P	Wiśniewski	W	Paweł P	Wiśniewski W
16	Paweł	P	Kowalski	W	Paweł P	Kowalski W
17	Paweł	P	Wójcik	W	Paweł P	Wójcik W
18	Paweł	P	Szymański	S	Paweł P	Szymański S
19	Paweł	P	Lewandowski	L	Paweł P	Lewandowski L
20	Michał	M	Wiśniewski	W	Michał M	Wiśniewski W
21	Michał	M	Lewandowski	L	Michał M	Lewandowski L
22	Michał	M	Kowalczyk	K	Michał M	Kowalczyk K
23	Michał	M	Szymański	S	Michał M	Szymański S
24	Michał	M	Kowalski	K	Michał M	Kowalski K
25	Jan	J	Lewandowski	L	Jan J	Lewandowski L

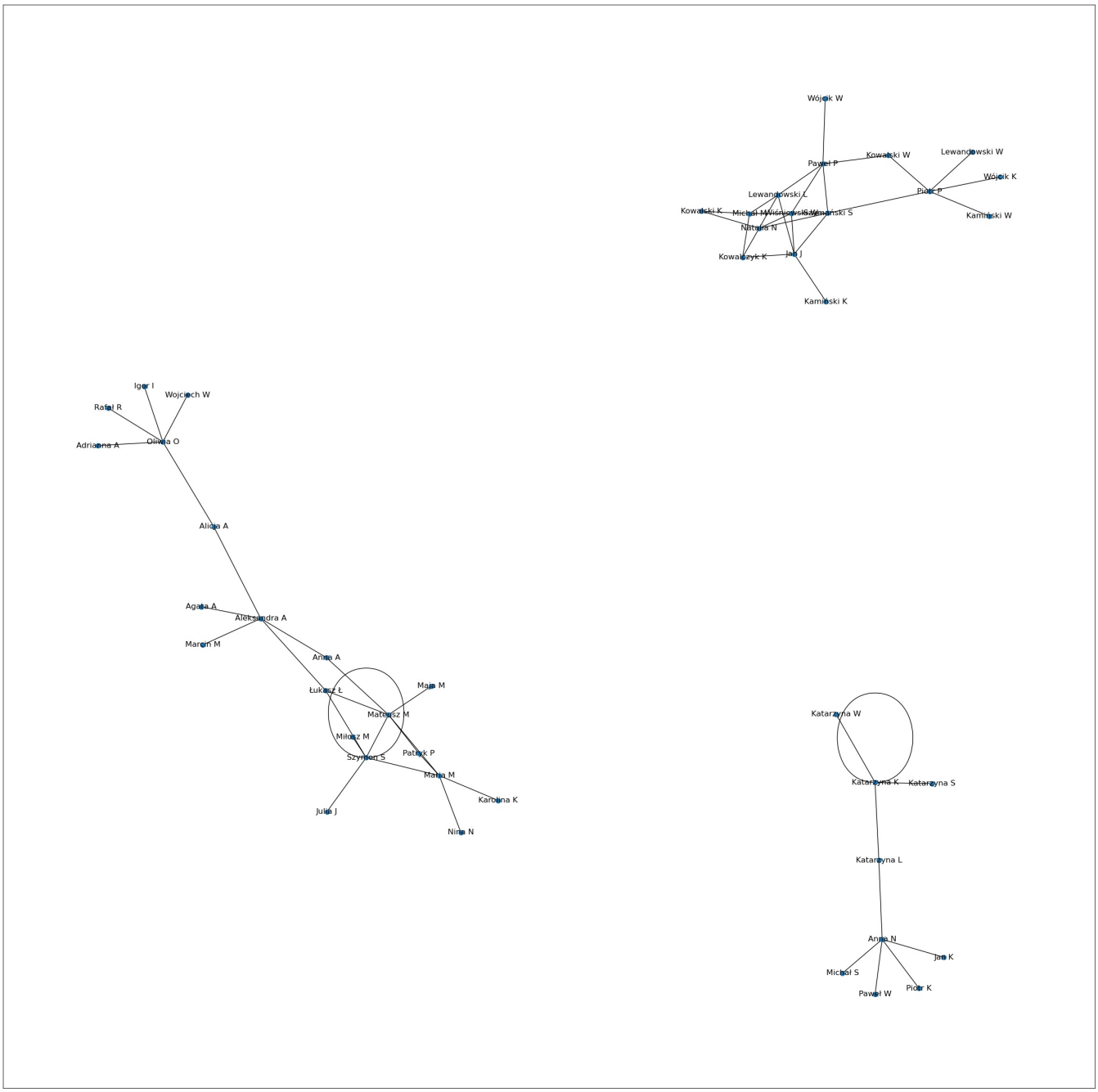
26	Jan	J	Wiśniewski	W	Jan J	Wiśniewski W
27	Jan	J	Kamiński	K	Jan J	Kamiński K
28	Jan	J	Szymański	S	Jan J	Szymański S
29	Jan	J	Kowalczyk	K	Jan J	Kowalczyk K
30	Natalia	N	Lewandowski	L	Natalia N	Lewandowski L
31	Natalia	N	Kowalski	K	Natalia N	Kowalski K
32	Natalia	N	Wiśniewski	W	Natalia N	Wiśniewski W
33	Natalia	N	Szymański	S	Natalia N	Szymański S
34	Natalia	N	Kowalczyk	K	Natalia N	Kowalczyk K
35	Mateusz	M	Łukasz	Ł	Mateusz M	Łukasz Ł
36	Mateusz	M	Szymon	S	Mateusz M	Szymon S
37	Mateusz	M	Patryk	P	Mateusz M	Patryk P
38	Mateusz	M	Anna	A	Mateusz M	Anna A
39	Mateusz	M	Maja	M	Mateusz M	Maja M
40	Aleksandra	A	Łukasz	Ł	Aleksandra A	Łukasz Ł
41	Aleksandra	A	Anna	A	Aleksandra A	Anna A
42	Aleksandra	A	Agata	A	Aleksandra A	Agata A
43	Aleksandra	A	Alicja	A	Aleksandra A	Alicja A
44	Aleksandra	A	Marcin	M	Aleksandra A	Marcin M
45	Maria	M	Karolina	K	Maria M	Karolina K
46	Maria	M	Mateusz	M	Maria M	Mateusz M
47	Maria	M	Szymon	S	Maria M	Szymon S
48	Maria	M	Nina	N	Maria M	Nina N
49	Maria	M	Patryk	P	Maria M	Patryk P
50	Szymon	S	Julia	J	Szymon S	Julia J
51	Szymon	S	Szymon	S	Szymon S	Szymon S
52	Szymon	S	Miłosz	M	Szymon S	Miłosz M
53	Szymon	S	Łukasz	Ł	Szymon S	Łukasz Ł
54	Szymon	S	Łukasz	Ł	Szymon S	Łukasz Ł
55	Oliwia	O	Adrianna	A	Oliwia O	Adrianna A
56	Oliwia	O	Wojciech	W	Oliwia O	Wojciech W
57	Oliwia	O	Igor	I	Oliwia O	Igor I
58	Oliwia	O	Rafał	R	Oliwia O	Rafał R
59	Oliwia	O	Alicja	A	Oliwia O	Alicja A

In [5]:
```
# prepare graph
graph = nx.from_pandas_edgelist(conections, source='Imien1', target='Imien2')
```

In [6]:
```
weight = nx.get_edge_attributes(graph,'Imię')
```

In [ ]:
```

```

In [7]:
```
plt.figure(figsize = (30,30))
nx.draw_networkx(graph,with_labels=True,node_size = 50)
plt.savefig("map_0.png", format = "png", dpi = 300)
plt.show()
```



In [8]:

```
# prepare node position
m = Basemap(projection='merc',llcrnrlon=16,llcrnrlat=48.4,urcrnrlon=18.2,
urcrnrlat=55.6, lat_ts=0, resolution='l',suppress_ticks=True)
mx, my = m(airports['Współrzędna geograficzna (długość)'].values, airports['Współrzędna geograficzna (szerokość
pos = {}
for count, elem in enumerate (airports['Imię']):
     pos[elem] = (mx[count], my[count])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   3628             try:
-> 3629                 return self._engine.get_loc(casted_key)
   3630             except KeyError as err:

~\anaconda3\lib\site-packages\pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

~\anaconda3\lib\site-packages\pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Współrzędna geograficzna (długość)'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_6820\699017054.py in <module>
      2 m = Basemap(projection='merc',llcrnrlon=16,llcrnrlat=48.4,urcrnrlon=18.2,
      3 urcrnrlat=55.6, lat_ts=0, resolution='l',suppress_ticks=True)
----> 4 mx, my = m(airports['Współrzędna geograficzna (długość)'].values, airports['Współrzędna geograficzna (s
zerokość)'].values)
      5 pos = {}
      6 for count, elem in enumerate (airports['Imię']):

~\anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
   3503             if self.columns.nlevels > 1:
   3504                 return self._getitem_multilevel(key)
-> 3505             indexer = self.columns.get_loc(key)
   3506             if is_integer(indexer):
   3507                 indexer = [indexer]

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   3629                 return self._engine.get_loc(casted_key)
   3630             except KeyError as err:
-> 3631                 raise KeyError(key) from err
   3632             except TypeError:
   3633                 # If we have a listlike key, _check_indexing_error will raise

KeyError: 'Współrzędna geograficzna (długość)'
```

In [ ]:
```
graph.nodes
```

In [ ]:
```
plt.figure(figsize = (10,10))
nx.draw_networkx(G = graph, pos = pos,with_labels=False,node_color = 'r',edge_color='b',alpha = 1, node_size =




plt.tight_layout()
plt.savefig("map_1.png", format = "png", dpi = 300)
plt.show()
plt.axis('off');
```

In [ ]:
```
start = input('Podaj stacje poczatkowa: ')
destination = input('Podaj stacje koncowa: ')

# start = 'Krakow'
# destination = 'Gdansk'

time = nx.shortest_path_length(graph, source=start, target=destination)
road = nx.shortest_path(graph, source=start, target=destination)

way = []
for i in range(len(road)):
    if i < len(road)-1:
        way.append((road[i],road[i+1]))

end = []
end.append(destination)

plt.figure(figsize = (10,10))
nx.draw_networkx(G = graph, pos = pos,with_labels=True,node_color = 'r',edge_color='y',alpha = 0.8, node_size =
nx.draw_networkx_edges(G = graph, pos = pos, edgelist=way,width = 3, edge_color='b')
nx.draw_networkx_nodes(G = graph, pos = pos, nodelist = end ,node_color = 'g',alpha = 0.8, node_size = 1000)

m.drawcountries(linewidth = 3)
m.drawstates(linewidth = 0.2)
plt.tight_layout()
plt.savefig("map_1.png", format = "png", dpi = 300)
plt.show()
plt.axis('off')

print(list(road))
```

```
print("Najkrótsza droga to {} \nOdległość między miastem {}, a miastem {} wynosi {} Minut.".format(road,start,de
```

In [8]:
```
graph.edges()
```

Out[8]:
```
EdgeView([('Anna N', 'Katarzyna L'), ('Anna N', 'Piotr K'), ('Anna N', 'Paweł W'), ('Anna N', 'Michał S'), ('An
na N', 'Jan K'), ('Katarzyna L', 'Katarzyna K'), ('Katarzyna K', 'Katarzyna W'), ('Katarzyna K', 'Katarzyna K')
, ('Katarzyna K', 'Katarzyna S'), ('Piotr P', 'Lewandowski W'), ('Piotr P', 'Kowalski W'), ('Piotr P', 'Kamińsk
i W'), ('Piotr P', 'Szymański S'), ('Piotr P', 'Wójcik K'), ('Kowalski W', 'Paweł P'), ('Szymański S', 'Paweł P
'), ('Szymański S', 'Michał M'), ('Szymański S', 'Jan J'), ('Szymański S', 'Natalia N'), ('Paweł P', 'Wiśniewsk
i W'), ('Paweł P', 'Wójcik W'), ('Paweł P', 'Lewandowski L'), ('Wiśniewski W', 'Michał M'), ('Wiśniewski W', 'J
an J'), ('Wiśniewski W', 'Natalia N'), ('Lewandowski L', 'Michał M'), ('Lewandowski L', 'Jan J'), ('Lewandowski
L', 'Natalia N'), ('Michał M', 'Kowalczyk K'), ('Michał M', 'Kowalski K'), ('Kowalczyk K', 'Jan J'), ('Kowalczy
k K', 'Natalia N'), ('Kowalski K', 'Natalia N'), ('Jan J', 'Kamiński K'), ('Mateusz M', 'Łukasz Ł'), ('Mateusz
M', 'Szymon S'), ('Mateusz M', 'Patryk P'), ('Mateusz M', 'Anna A'), ('Mateusz M', 'Maja M'), ('Mateusz M', 'Ma
ria M'), ('Łukasz Ł', 'Aleksandra A'), ('Łukasz Ł', 'Szymon S'), ('Szymon S', 'Maria M'), ('Szymon S', 'Julia J
'), ('Szymon S', 'Szymon S'), ('Szymon S', 'Miłosz M'), ('Patryk P', 'Maria M'), ('Anna A', 'Aleksandra A'), ('
Aleksandra A', 'Agata A'), ('Aleksandra A', 'Alicja A'), ('Aleksandra A', 'Marcin M'), ('Alicja A', 'Oliwia O')
, ('Maria M', 'Karolina K'), ('Maria M', 'Nina N'), ('Oliwia O', 'Adrianna A'), ('Oliwia O', 'Wojciech W'), ('O
liwia O', 'Igor I'), ('Oliwia O', 'Rafał R')])
```

In [9]:
```
print (nx.info(graph))
```

```
Graph with 45 nodes and 58 edges
```

```
C:\Users\wlabl\AppData\Local\Temp\ipykernel_15984\944726865.py:1: DeprecationWarning: info is deprecated and wi
ll be removed in version 3.0.

  print (nx.info(graph))
```

In [8]:
```
graph.number_of_nodes()
```

Out[8]:
```
45
```

In [9]:
```
graph.describe()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_13736\117657790.py in <module>
----> 1 graph.describe()

AttributeError: 'Graph' object has no attribute 'describe'
```

In [ ]:
```
graph.number_of_edges()
```

In [ ]:
```
sorted(graph.degree, key=lambda x: x[1], reverse=True)
```

In [ ]:
```
G = nx.Graph([ ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
```

```
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A")])
for C in (G.subgraph(c).copy() for c in nx.connected_components(G)):
    print(nx.average_clustering(C))
```

In [10]:
```
nx.average_clustering(graph)
```

Out[10]:
```
0.04296296296296297
```

In [11]:
```
nx.global_efficiency(graph)
```

Out[11]:
```
0.15785473785473808
```

In [12]:
```
list(graph.nodes)
```

Out[12]:
```
['Anna N',
 'Katarzyna L',
 'Piotr K',
 'Paweł W',
 'Michał S',
 'Jan K',
 'Katarzyna K',
 'Katarzyna W',
 'Katarzyna S',
 'Piotr P',
 'Lewandowski W',
 'Kowalski W',
 'Kamiński W',
 'Szymański S',
 'Wójcik K',
 'Paweł P',
 'Wiśniewski W',
 'Wójcik W',
 'Lewandowski L',
 'Michał M',
 'Kowalczyk K',
 'Kowalski K',
 'Jan J',
 'Kamiński K',
 'Natalia N',
 'Mateusz M',
 'Łukasz Ł',
 'Szymon S',
 'Patryk P',
 'Anna A',
 'Maja M',
 'Aleksandra A',
 'Agata A',
 'Alicja A',
 'Marcin M',
 'Maria M',
 'Karolina K',
 'Nina N',
 'Julia J',
 'Miłosz M',
 'Oliwia O',
 'Adrianna A',
 'Wojciech W',
 'Igor I',
 'Rafał R']
```

In [13]:
```
import numpy as np
np.set_printoptions(linewidth=110)
A = nx.adjacency_matrix(graph)
print(A.todense())
```

```
C:\Users\wlabl\AppData\Local\Temp\ipykernel_13736\3929157849.py:3: FutureWarning: adjacency_matrix will return
a scipy.sparse array instead of a matrix in Networkx 3.0.
  A = nx.adjacency_matrix(graph)
C:\Users\wlabl\anaconda3\lib\site-packages\scipy\__init__.py:177: UserWarning: A NumPy version >=1.18.5 and <1.
26.0 is required for this version of SciPy (detected version 1.26.4
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
[[0 1 1 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

In [14]:
```
nx.degree_pearson_correlation_coefficient(graph)
```

Out[14]:
```
-0.5101760176017602
```

In [15]:
```
G = nx.Graph([ ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A")])
for C in (G.subgraph(c).copy() for c in nx.connected_components(G)):
    print(nx.diameter(C))
```

```
4
4
7
```

In [16]:
```
nx.global_efficiency(graph)
```

Out[16]:
```
0.15785473785473808
```

In [17]:
```
G = nx.Graph([ ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A")])
for C in (G.subgraph(c).copy() for c in nx.connected_components(G)):
    print(nx.average_shortest_path_length(C))
```

```
2.388888888888889
2.4583333333333335
3.405263157894737
```

In [18]:
```
nx.average_shortest_path_length(G)
```

```
---------------------------------------------------------------------------
NetworkXError                             Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_13736\3504744090.py in <module>
----> 1 nx.average_shortest_path_length(G)

~\anaconda3\lib\site-packages\networkx\algorithms\shortest_paths\generic.py in average_shortest_path_length(G,
weight, method)
    402         raise nx.NetworkXError("Graph is not weakly connected.")
    403     if not G.is_directed() and not nx.is_connected(G):
--> 404         raise nx.NetworkXError("Graph is not connected.")
    405
    406     # Compute all-pairs shortest paths.

NetworkXError: Graph is not connected.
```

In [ ]:
```
G = nx.Graph([("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
```

```
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A")])
for C in (G.subgraph(c).copy() for c in nx.connected_components(G)):
    print(nx.average_shortest_path_length(C))
```

In [ ]:

```
import matplotlib.pyplot as plt
from matplotlib.cm import ScalarMappable
import networkx as nx

g = nx.Graph([("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
```

```
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A")])
gc = g.subgraph(max(nx.connected_components(g)))
lcc = nx.clustering(gc)

cmap = plt.get_cmap('winter')
norm = plt.Normalize(0, max(lcc.values()))
node_colors = [cmap(norm(lcc[node])) for node in gc.nodes]

fig, (ax1) = plt.subplots(ncols=1, figsize=(30, 20))
nx.draw_spring(gc, node_color=node_colors,node_size = 200, with_labels=True, ax=ax1)
fig.colorbar(ScalarMappable(cmap=cmap, norm=norm), label='Clustering', shrink=0.95, ax=ax1)

ax2.hist(lcc.values(), bins=10)
ax2.set_xlabel('Clustering')
ax2.set_ylabel('Frequency')

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

In [19]:
```
network_graph = graph.gamma(network_graph, radius=3)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_13736\1591016959.py in <module>
----> 1 network_graph = graph.gamma(network_graph, radius=3)

AttributeError: 'Graph' object has no attribute 'gamma'
```

In [20]:
```
nx.average_node_connectivity(graph, flow_func=None)
```

Out[20]:
```
0.47474747474747475
```

In [21]:
```
nx.average_shortest_path_length(graph)
```

```
---------------------------------------------------------------------------
NetworkXError                             Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_13736\1334074829.py in <module>
----> 1 nx.average_shortest_path_length(graph)

~\anaconda3\lib\site-packages\networkx\algorithms\shortest_paths\generic.py in average_shortest_path_length(G,
weight, method)
    402         raise nx.NetworkXError("Graph is not weakly connected.")
    403     if not G.is_directed() and not nx.is_connected(G):
--> 404         raise nx.NetworkXError("Graph is not connected.")
    405
    406     # Compute all-pairs shortest paths.

NetworkXError: Graph is not connected.
```

In [38]:
```
nx.diameter(graph, e=None, usebounds=False)
```

```
---------------------------------------------------------------------------
NetworkXError                             Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_15996\3745513747.py in <module>
----> 1 nx.diameter(graph, e=None, usebounds=False)

~\anaconda3\lib\site-packages\networkx\algorithms\distance_measures.py in diameter(G, e, usebounds)
    386         return _extrema_bounding(G, compute="diameter")
    387     if e is None:
--> 388         e = eccentricity(G)
    389     return max(e.values())
    390

~\anaconda3\lib\site-packages\networkx\algorithms\distance_measures.py in eccentricity(G, v, sp)
    345             else:
    346                 msg = "Found infinite path length because the graph is not" " connected"
--> 347             raise nx.NetworkXError(msg)
    348
    349         e[n] = max(length.values())

NetworkXError: Found infinite path length because the graph is not connected
```

In [39]:
```
G = nx.Graph([('Anna K', 'Jan W'), ('Anna K', 'Maria K'), ('Anna K', 'Piotr M'), ('Anna K', 'Katarzyna K'), ('A
for C in (G.subgraph(c).copy() for c in nx.connected_components(G)):
    print(nx.average_shortest_path_length(C))
```

```
1.8571428571428572
```

In [41]:
```
nx.betweenness_centrality(graph, k=None, normalized=True, weight=None, endpoints=False, seed=None)
```

Out[41]:
```
{'Anna N': 0.023255813953488372,
 'Katarzyna L': 0.015856236786469344,
 'Piotr K': 0.0,
 'Paweł W': 0.0,
 'Michał S': 0.0,
 'Jan K': 0.0,
 'Katarzyna K': 0.013742071881606767,
 'Katarzyna W': 0.0,
 'Katarzyna S': 0.0,
 'Piotr P': 0.04334038054968288,
 'Lewandowski W': 0.0,
 'Kowalski W': 0.005919661733615223,
 'Kamiński W': 0.0,
 'Szymański S': 0.0455602536997886,
 'Wójcik K': 0.0,
 'Paweł P': 0.026374207188160678,
 'Wiśniewski W': 0.007610993657505285,
 'Wójcik W': 0.0,
 'Lewandowski L': 0.007610993657505285,
 'Michał M': 0.01287878787878788,
 'Kowalczyk K': 0.0017970401691331927,
 'Kowalski K': 0.00021141649048625794,
 'Jan J': 0.020806906272022554,
 'Kamiński K': 0.0,
 'Natalia N': 0.01287878787878788,
 'Mateusz M': 0.05972515856236787,
 'Łukasz Ł': 0.06183932346723044,
 'Szymon S': 0.04809725158562368,
 'Patryk P': 0.0,
 'Anna A': 0.02378435517970402,
 'Maja M': 0.0,
 'Aleksandra A': 0.1072938689217759,
 'Agata A': 0.0,
 'Alicja A': 0.07399577167019028,
 'Marcin M': 0.0,
 'Maria M': 0.03858350951374207,
 'Karolina K': 0.0,
 'Nina N': 0.0,
 'Julia J': 0.0,
 'Miłosz M': 0.0,
 'Oliwia O': 0.06976744186046512,
 'Adrianna A': 0.0,
 'Wojciech W': 0.0,
 'Igor I': 0.0,
 'Rafał R': 0.0}
```

In [16]:
```
conections.describe()
```

Out[16]:

	column1	Column2	column2	Column4	Imien1	Imien2
count	60	60	60	60	60	60
unique	12	8	29	12	12	36
top	Anna	M	Katarzyna	K	Anna N	Szymański S
freq	5	15	6	12	5	5

In [29]:
```
nx.draw(graph, pos, nx.degree_centrality(graph), 'Degree Centrality')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_7304\327545051.py in <module>
----> 1 nx.draw(graph, pos, nx.degree_centrality(graph), 'Degree Centrality')

TypeError: draw() takes from 1 to 3 positional arguments but 4 were given
```

In [30]:
```
nx.draw(graph, pos, nx.eigenvector_centrality(graph), 'Eigenvector Centrality')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_7304\1790819775.py in <module>
----> 1 nx.draw(graph, pos, nx.eigenvector_centrality(graph), 'Eigenvector Centrality')

TypeError: draw() takes from 1 to 3 positional arguments but 4 were given
```

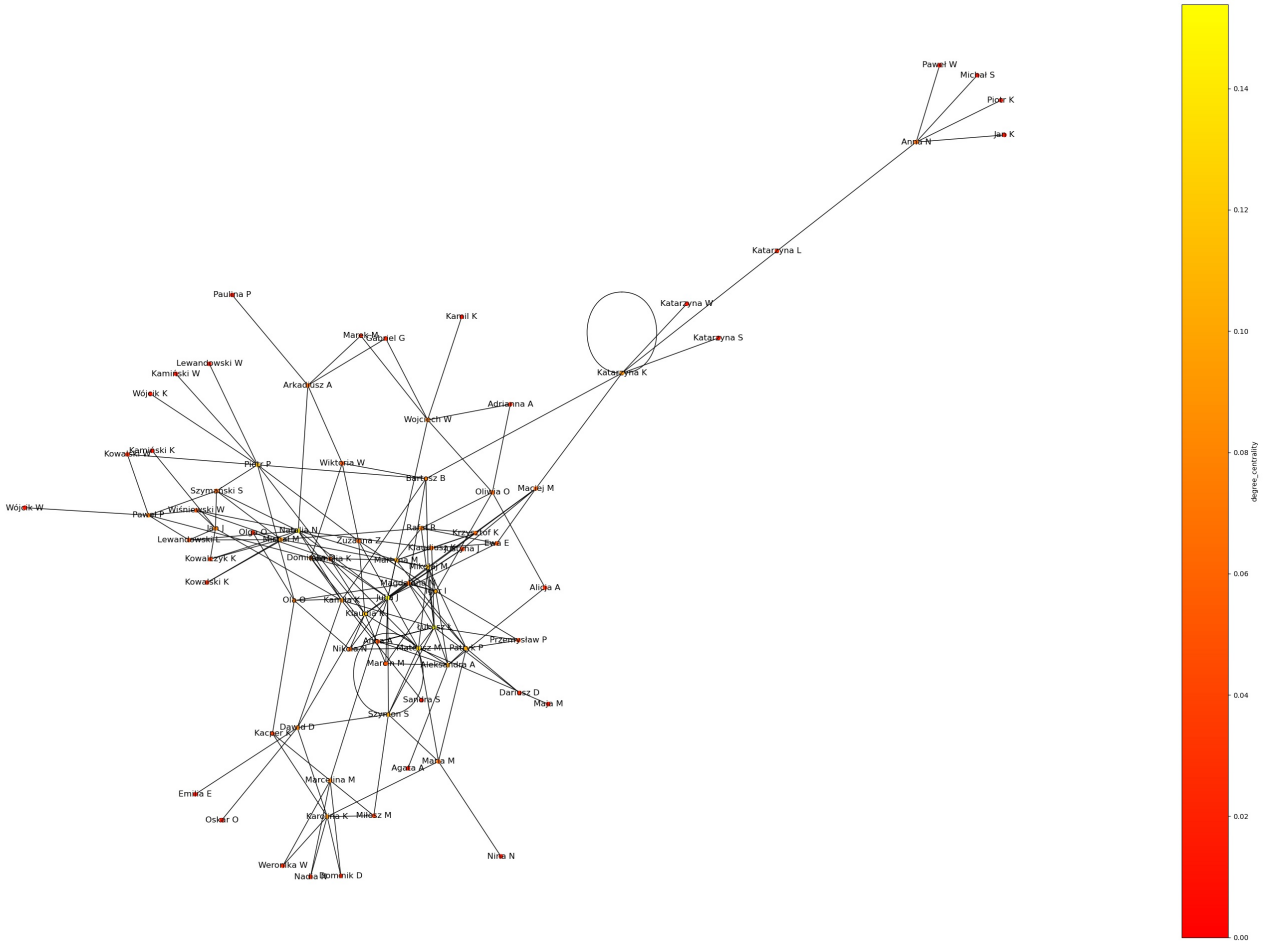
In [31]:
```
nx.draw(graph, pos, nx.katz_centrality(graph, alpha=0.1, beta=1.0), 'Katz Centrality')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_7304\4087637726.py in <module>
----> 1 nx.draw(graph, pos, nx.katz_centrality(graph, alpha=0.1, beta=1.0), 'Katz Centrality')

TypeError: draw() takes from 1 to 3 positional arguments but 4 were given
```

In [32]:
```
nx.draw(graph, pos, nx.betweenness_centrality(graph), 'Betweenness Centrality')
```

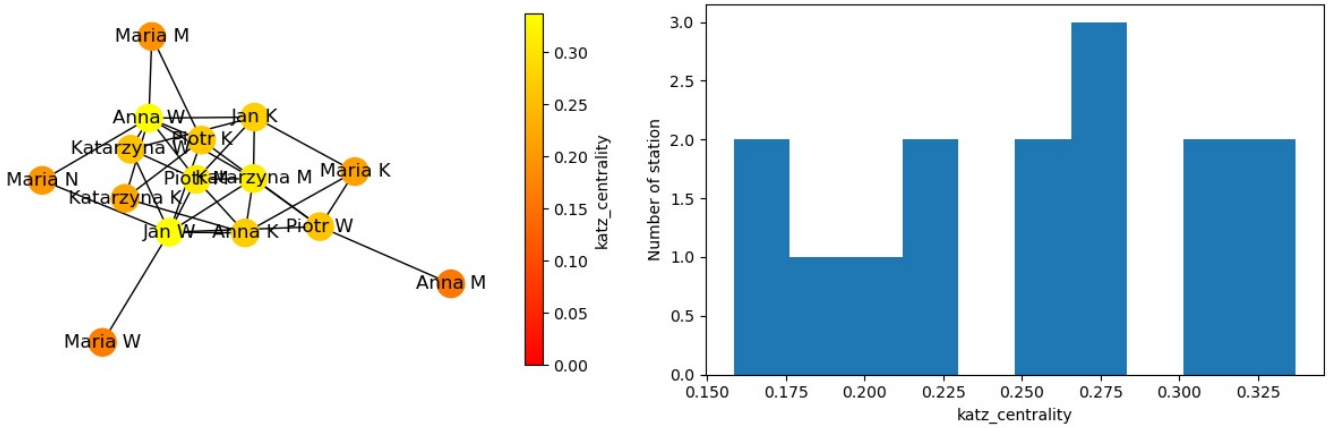
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_7304\2815298285.py in <module>
----> 1 nx.draw(graph, pos, nx.betweenness_centrality(graph), 'Betweenness Centrality')

TypeError: draw() takes from 1 to 3 positional arguments but 4 were given
```

In [27]:
```
import matplotlib.pyplot as plt
from matplotlib.cm import ScalarMappable
import networkx as nx

g = nx.Graph([('Anna N', 'Katarzyna L'), ('Anna N', 'Piotr K'), ('Anna N', 'Paweł W'), ('Anna N', 'Michał S'),
gc = g.subgraph(max(nx.connected_components(g)))
lcc = nx.degree_centrality(gc)

cmap = plt.get_cmap('')
norm = plt.Normalize(0, max(lcc.values()))
node_colors = [cmap(norm(lcc[node])) for node in gc.nodes]

fig, (ax1) = plt.subplots(ncols=1, figsize=(30, 20))
nx.draw_spring(gc, node_color=node_colors, node_size=30,with_labels=True, ax=ax1)
fig.colorbar(ScalarMappable(cmap=cmap, norm=norm), label='degree_centrality', shrink=0.95, ax=ax1)

ax2.hist(lcc.values(), bins=10)
ax2.set_xlabel('degree_centrality')
ax2.set_ylabel('Number of station')

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



In [34]:

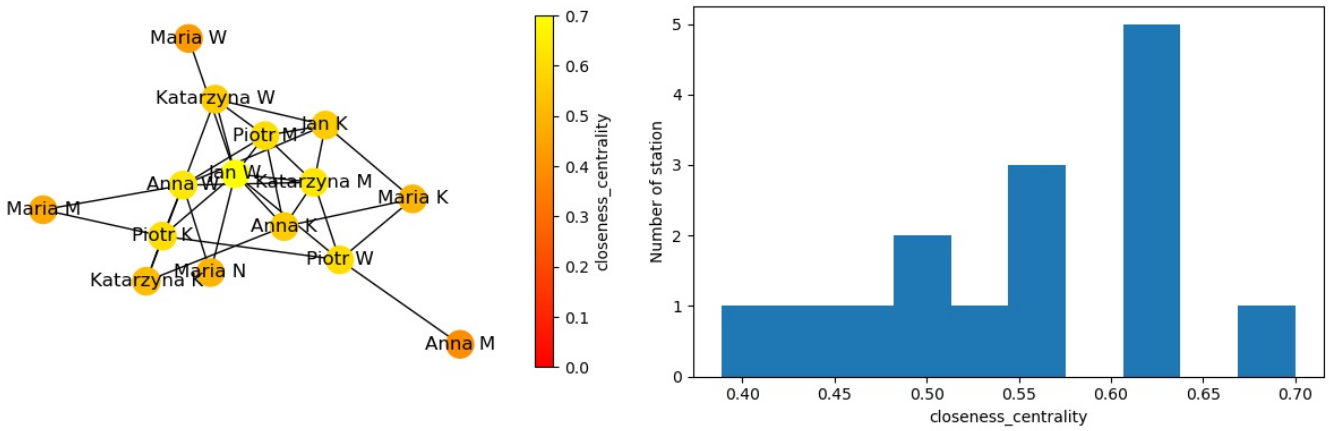
```
import matplotlib.pyplot as plt
from matplotlib.cm import ScalarMappable
import networkx as nx

g = nx.Graph([('Anna K', 'Jan W'), ('Anna K', 'Maria K'), ('Anna K', 'Piotr M'), ('Anna K', 'Katarzyna K'), ('A
gc = g.subgraph(max(nx.connected_components(g)))
lcc = nx.katz_centrality(gc)

cmap = plt.get_cmap('autumn')
norm = plt.Normalize(0, max(lcc.values()))
node_colors = [cmap(norm(lcc[node])) for node in gc.nodes]

fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12, 4))
nx.draw_spring(gc, node_color=node_colors, with_labels=True, ax=ax1)
fig.colorbar(ScalarMappable(cmap=cmap, norm=norm), label='katz_centrality', shrink=0.95, ax=ax1)

ax2.hist(lcc.values(), bins=10)
ax2.set_xlabel('katz_centrality')
ax2.set_ylabel('Number of station')

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



In [35]:

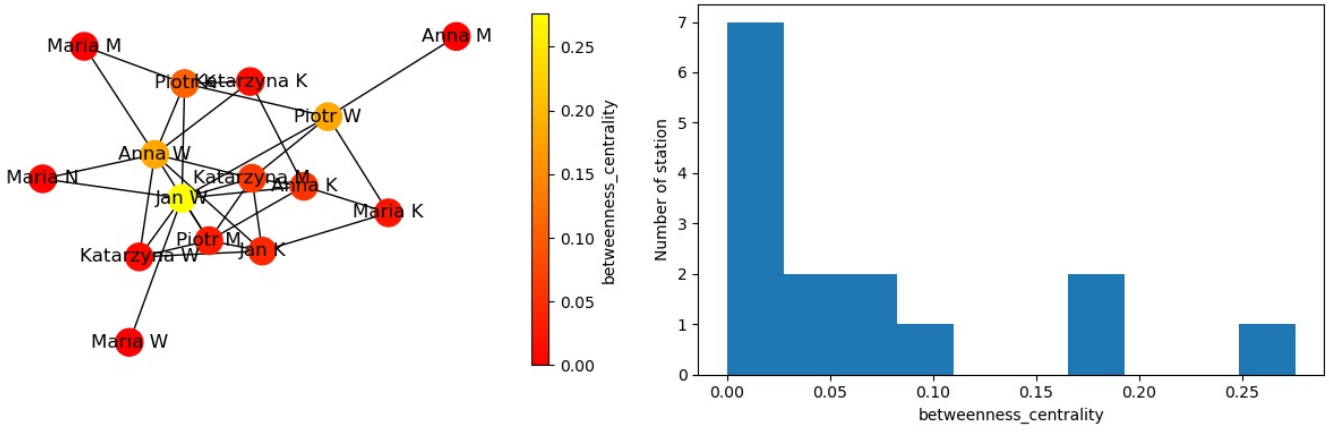
```
import matplotlib.pyplot as plt
from matplotlib.cm import ScalarMappable
import networkx as nx

g = nx.Graph([('Anna K', 'Jan W'), ('Anna K', 'Maria K'), ('Anna K', 'Piotr M'), ('Anna K', 'Katarzyna K'), ('A
```

```
gc = g.subgraph(max(nx.connected_components(g)))
lcc = nx.closeness_centrality(gc)

cmap = plt.get_cmap('autumn')
norm = plt.Normalize(0, max(lcc.values()))
node_colors = [cmap(norm(lcc[node])) for node in gc.nodes]

fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12, 4))
nx.draw_spring(gc, node_color=node_colors, with_labels=True, ax=ax1)
fig.colorbar(ScalarMappable(cmap=cmap, norm=norm), label='closeness_centrality', shrink=0.95, ax=ax1)

ax2.hist(lcc.values(), bins=10)
ax2.set_xlabel('closeness_centrality')
ax2.set_ylabel('Number of station')

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



In [36]:

```
import matplotlib.pyplot as plt
from matplotlib.cm import ScalarMappable
import networkx as nx

g = nx.Graph([('Anna K', 'Jan W'), ('Anna K', 'Maria K'), ('Anna K', 'Piotr M'), ('Anna K', 'Katarzyna K'), ('A
gc = g.subgraph(max(nx.connected_components(g)))
lcc = nx.betweenness_centrality(gc)

cmap = plt.get_cmap('autumn')
norm = plt.Normalize(0, max(lcc.values()))
node_colors = [cmap(norm(lcc[node])) for node in gc.nodes]

fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12, 4))
nx.draw_spring(gc, node_color=node_colors, with_labels=True, ax=ax1)
fig.colorbar(ScalarMappable(cmap=cmap, norm=norm), label='betweenness_centrality', shrink=0.95, ax=ax1)

ax2.hist(lcc.values(), bins=10)
ax2.set_xlabel('betweenness_centrality')
ax2.set_ylabel('Number of station')

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



In [37]:

```
nx.to_pandas_adjacency(graph)
```

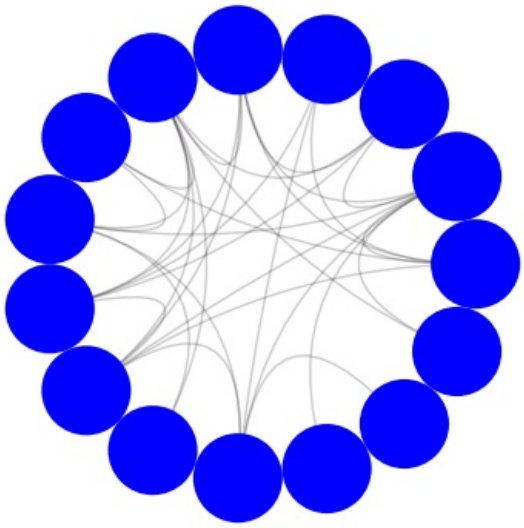
Out[37]:

	Anna K	Jan W	Katarzyna W	Maria K	Piotr M	Anna W	Katarzyna K	Piotr K	Jan K	Katarzyna M	Maria M	Piotr W	Maria W	Anna M	Maria N
Anna K	0.0	1.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
Jan W	1.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0
Katarzyna W	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
Maria K	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0
Piotr M	1.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0
Anna W	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
Katarzyna K	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Piotr K	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0
Jan K	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
Katarzyna M	1.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0
Maria M	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Piotr W	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0
Maria W	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Anna M	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
Maria N	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

In [38]:
```
import nxviz as nv
nv.circos(graph)
```

```
C:\Users\wlabl\anaconda3\lib\site-packages\nxviz\__init__.py:18: UserWarning:
nxviz has a new API! Version 0.7.4 onwards, the old class-based API is being
deprecated in favour of a new API focused on advancing a grammar of network
graphics. If your plotting code depends on the old API, please consider
pinning nxviz at version 0.7.4, as the new API will break your old code.

To check out the new API, please head over to the docs at
https://ericmjl.github.io/nxviz/ to learn more. We hope you enjoy using it!

(This deprecation message will go away in version 1.0.)

  warnings.warn(
```

Out[38]:
```
<AxesSubplot:>
```



In [39]:
```
nx.average_degree_connectivity(graph)
```

Out[39]:
```
{5: 5.1, 8: 4.1875, 4: 6.75, 3: 5.5, 6: 6.083333333333333, 2: 7.25, 1: 6.5}
```

In [40]:
```
nx.average_shortest_path_length(graph)
```

Out[40]:
```
1.8571428571428572
```

In [41]:
```
G = nx.Graph([('Vorosmarty square M1 ', 'Deak Square M1'), ('Deak Square M1', 'Bajcsy-Zsilinszky Way M1  '), ('
for C in (G.subgraph(c).copy() for c in nx.connected_components(G)):
    print(nx.average_shortest_path_length(C))
```

```
7.605911330049261
3.6666666666666665
3.6666666666666665
```

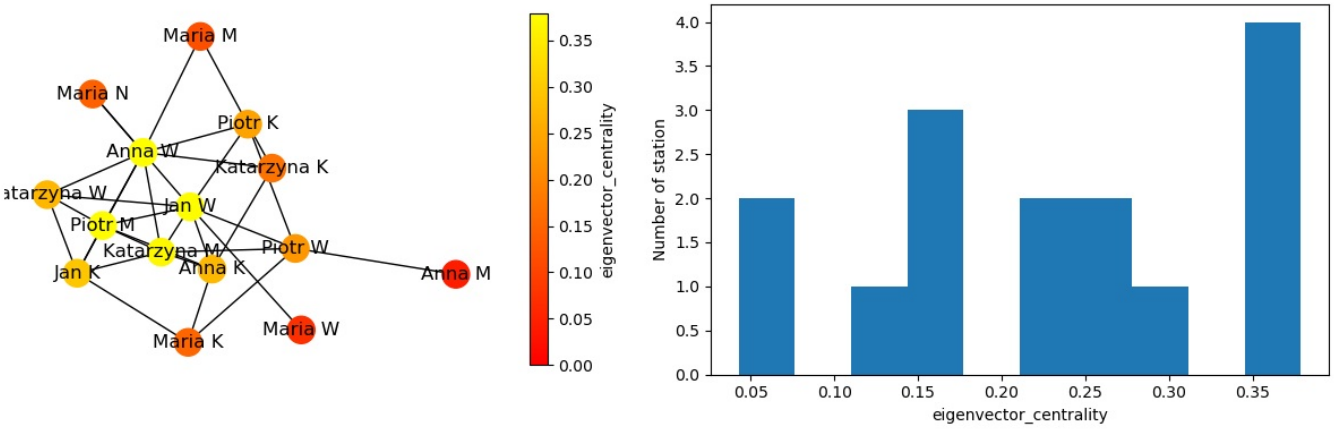
In [42]:
```
nx.density(graph)
```

Out[42]:
```
0.3047619047619048
```

In [43]:
```
nx.global_efficiency(graph)
```

Out[43]:
```
0.6253968253968251
```

In [44]:
```
import matplotlib.pyplot as plt
from matplotlib.cm import ScalarMappable
import networkx as nx

g = nx.Graph([('Anna K', 'Jan W'), ('Anna K', 'Maria K'), ('Anna K', 'Piotr M'), ('Anna K', 'Katarzyna K'), ('A
gc = g.subgraph(max(nx.connected_components(g)))
lcc = nx.eigenvector_centrality(gc)

cmap = plt.get_cmap('autumn')
norm = plt.Normalize(0, max(lcc.values()))
node_colors = [cmap(norm(lcc[node])) for node in gc.nodes]

fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12, 4))
nx.draw_spring(gc, node_color=node_colors, with_labels=True, ax=ax1)
fig.colorbar(ScalarMappable(cmap=cmap, norm=norm), label='eigenvector_centrality', shrink=0.95, ax=ax1)

ax2.hist(lcc.values(), bins=10)
ax2.set_xlabel('eigenvector_centrality')
ax2.set_ylabel('Number of station')

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



In [45]:
```
nx.louvain_communities(graph, resolution=1, threshold=1e-07, seed=None)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_7304\2989943730.py in <module>
----> 1 nx.louvain_communities(graph, resolution=1, threshold=1e-07, seed=None)

~\anaconda3\lib\site-packages\networkx\__init__.py in __getattr__(name)
     49             "This message will be removed in NetworkX 3.0."
     50         )
---> 51     raise AttributeError(f"module {__name__} has no attribute {name}")
     52
     53

AttributeError: module networkx has no attribute louvain_communities
```

In [53]:
```
import networkx as nx
G = nx.petersen_graph()
nx.community.louvain_communities(G, seed=123)
[('Anna K', 'Jan W'), ('Anna K', 'Maria K'), ('Anna K', 'Piotr M'), ('Anna K', 'Katarzyna K'), ('Anna K', 'Kata
```

Out[53]:
```
[('Anna K', 'Jan W'),
 ('Anna K', 'Maria K'),
 ('Anna K', 'Piotr M'),
 ('Anna K', 'Katarzyna K'),
 ('Anna K', 'Katarzyna M'),
 ('Jan W', 'Katarzyna W'),
 ('Jan W', 'Piotr K'),
 ('Jan W', 'Piotr M'),
 ('Jan W', 'Maria W'),
 ('Jan W', 'Maria N'),
 ('Jan W', 'Katarzyna M'),
 ('Jan W', 'Piotr W'),
 ('Katarzyna W', 'Piotr M'),
 ('Katarzyna W', 'Jan K'),
 ('Katarzyna W', 'Anna W'),
 ('Maria K', 'Jan K'),
 ('Maria K', 'Piotr W'),
 ('Piotr M', 'Anna W'),
 ('Piotr M', 'Jan K'),
 ('Piotr M', 'Katarzyna M'),
 ('Anna W', 'Katarzyna K'),
 ('Anna W', 'Katarzyna M'),
 ('Anna W', 'Maria M'),
 ('Anna W', 'Jan K'),
 ('Anna W', 'Maria N'),
 ('Anna W', 'Piotr K'),
 ('Katarzyna K', 'Piotr K'),
 ('Piotr K', 'Maria M'),
 ('Piotr K', 'Piotr W'),
 ('Jan K', 'Katarzyna M'),
 ('Katarzyna M', 'Piotr W'),
 ('Piotr W', 'Anna M')]
```

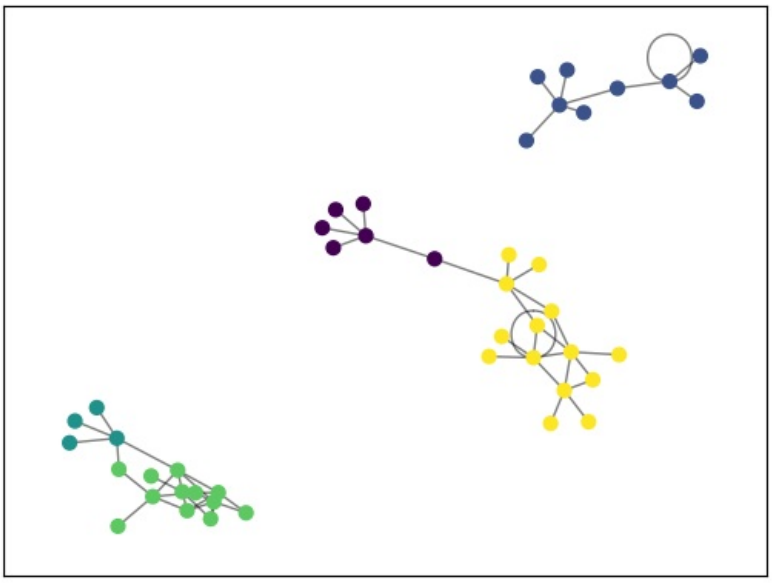
In [46]:
```
import community as community_louvain
>>> import networkx as nx
>>> G = nx.erdos_renyi_graph(100, 0.01)
>>> partion = community_louvain.best_partition(G)
```

In [65]:
```
>>> # display a graph with its communities:
>>> # as Erdos-Renyi graphs don't have true community structure,
>>> # instead load the karate club graph
>>> import community as community_louvain
>>> import matplotlib.cm as cm
>>> import matplotlib.pyplot as plt
>>> import networkx as nx
>>>
g = nx.Graph([("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
```

```
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A")
])
>>> # compute the best partition
>>> partition = community_louvain.best_partition(g)
```

In [66]:

```
>>> # draw the graph
>>> pos = nx.spring_layout(g)
>>> # color the nodes according to their partition
>>> cmap = cm.get_cmap('viridis', max(partition.values()) + 1)
>>> nx.draw_networkx_nodes(g, pos, partition.keys(), node_size=40,
>>>                        cmap=cmap, node_color=list(partition.values()))
>>> nx.draw_networkx_edges(g, pos,  alpha=0.5)
>>> plt.show()
```



In [61]:

```
import networkx as nx
import math

def calculate_entropy(probabilities):
    entropy = -sum(p * math.log2(p) for p in probabilities if p != 0)
    return entropy

def main():
    # Create a sample graph using NetworkX
    G = nx.Graph()
    G.add_edges_from([("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
```

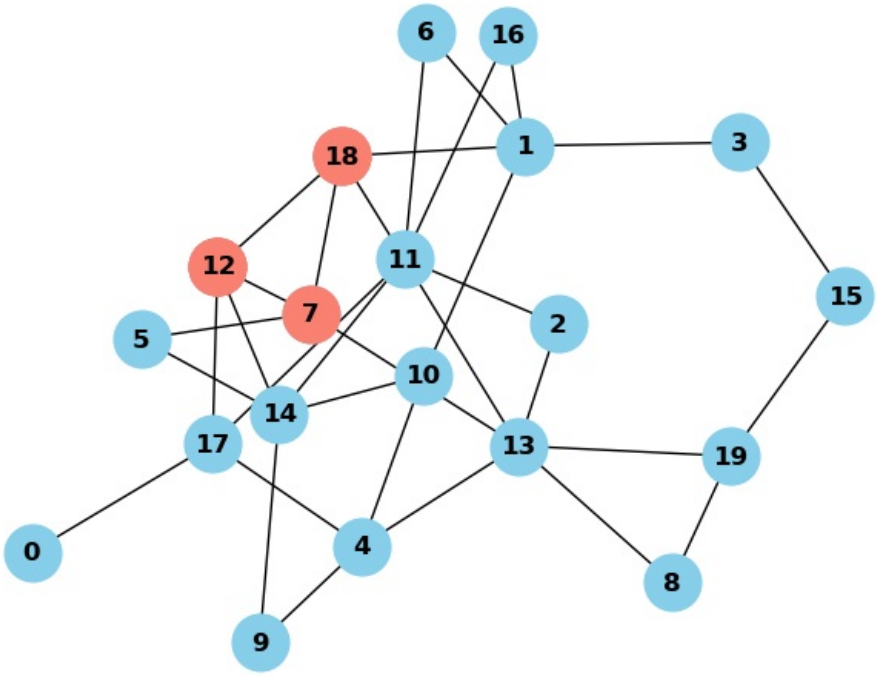
```
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A")
])

    # Calculate probabilities for each node's degree
    degree_probabilities = [G.degree(node) / len(G) for node in G.nodes()]

    # Calculate entropy based on node degrees
    entropy = calculate_entropy(degree_probabilities)
    print(f"Entropy of the graph: {entropy}")

    # Perform additional analysis using NetworkX or Shannon's Information Theory concepts

if __name__ == "__main__":
    main()
```

```
Entropy of the graph: 9.703952877608451
```

In [41]:

```
import networkx as nx
import matplotlib.pyplot as plt
import random

def random_walk(graph, start_node, num_steps):
    current_node = start_node
    visited_nodes = [current_node]

    for _ in range(num_steps):
        neighbors = list(graph.neighbors(current_node))
        if not neighbors:
            break  # Break if the current node has no neighbors

        next_node = random.choice(neighbors)
        visited_nodes.append(next_node)
        current_node = next_node

    return visited_nodes

def visualize_graph(graph, path):
    pos = nx.spring_layout(graph)
    nx.draw(graph, pos, with_labels=True, node_color='skyblue', font_weight='bold', node_size=700)
    nx.draw_networkx_nodes(graph, pos, nodelist=path, node_color='salmon', node_size=700)
    plt.show()

def main():
    # Create a sample graph
    G = nx.erdos_renyi_graph(20, 0.2)

    # Choose a starting node for the random walk
    start_node = random.choice(list(G.nodes))

    # Simulate information flow through a random walk
    num_steps = 5
    path = random_walk(G, start_node, num_steps)
```

```
    # Visualize the graph with the path of the random walk highlighted
    visualize_graph(G, path)

if __name__ == "__main__":
    main()
```



In [31]:
```
nx.node_redundancy(G, nodes=None)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_9024\1467277173.py in <module>
----> 1 nx.node_redundancy(G, nodes=None)

~\anaconda3\lib\site-packages\networkx\__init__.py in __getattr__(name)
     49             "This message will be removed in NetworkX 3.0."
     50         )
---> 51     raise AttributeError(f"module {__name__} has no attribute {name}")
     52
     53

AttributeError: module networkx has no attribute node_redundancy
```

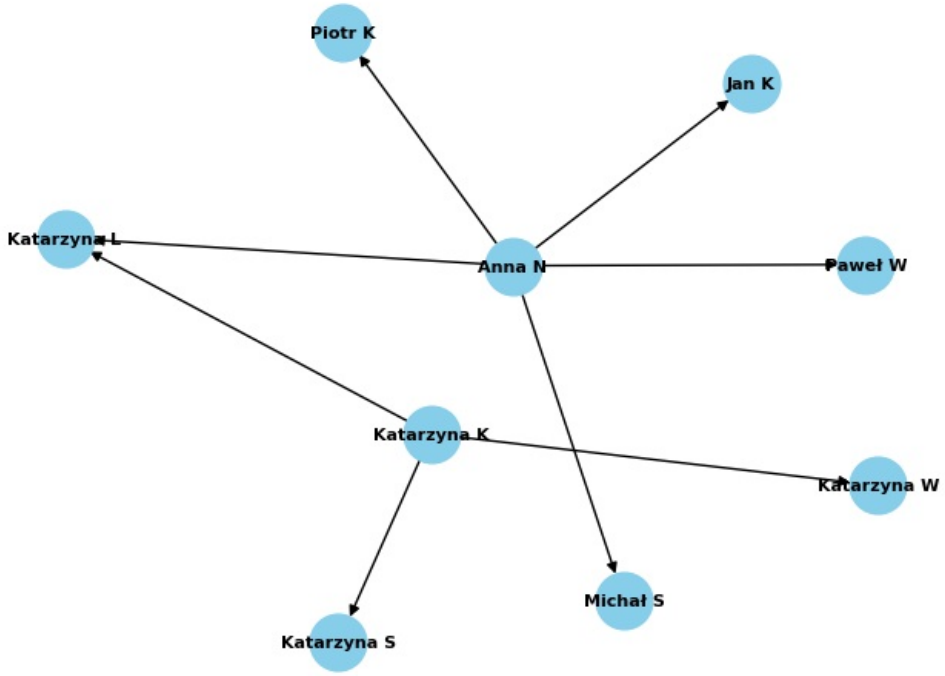
In [41]:
```
from itertools import combinations

import networkx as nx
from networkx import NetworkXError

__all__ = ["node_redundancy"]
```

In [45]:
```
import networkx as nx

# Tworzenie skierowanego grafu
G = nx.DiGraph()

# Dodawanie krawędzi z wagami (pojemnościami)
G.add_edge("Źródło", "A", capacity=3)
G.add_edge("Źródło", "B", capacity=2)
G.add_edge("A", "C", capacity=3)
G.add_edge("B", "C", capacity=2)
G.add_edge("B", "Koniec", capacity=1)
G.add_edge("C", "Koniec", capacity=3)

# Obliczanie maksymalnego przepływu
max_flow_value, max_flow_dict = nx.maximum_flow(G, "Źródło", "Koniec")

print("Maksymalny przepływ:", max_flow_value)
print("Przepływ przez krawędzie:", max_flow_dict)
```

```
Maksymalny przepływ: 4
Przepływ przez krawędzie: {'Źródło': {'A': 2, 'B': 2}, 'A': {'C': 2}, 'B': {'C': 1, 'Koniec': 1}, 'C': {'Koniec
': 3}, 'Koniec': {}}
```

In [2]:
```
import networkx as nx
import matplotlib.pyplot as plt
```

```
# Tworzenie skierowanego grafu
G = nx.DiGraph()

# Dodawanie krawędzi z wcześniejszych połączeń
edges = [
    ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna S"),

    # Dodaj pozostałe połączenia
]

G.add_edges_from(edges)

# Rysowanie grafu
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, font_weight='bold', node_size=700, node_color='skyblue', font_color='black',

# Wyświetlanie grafu
plt.show()
```

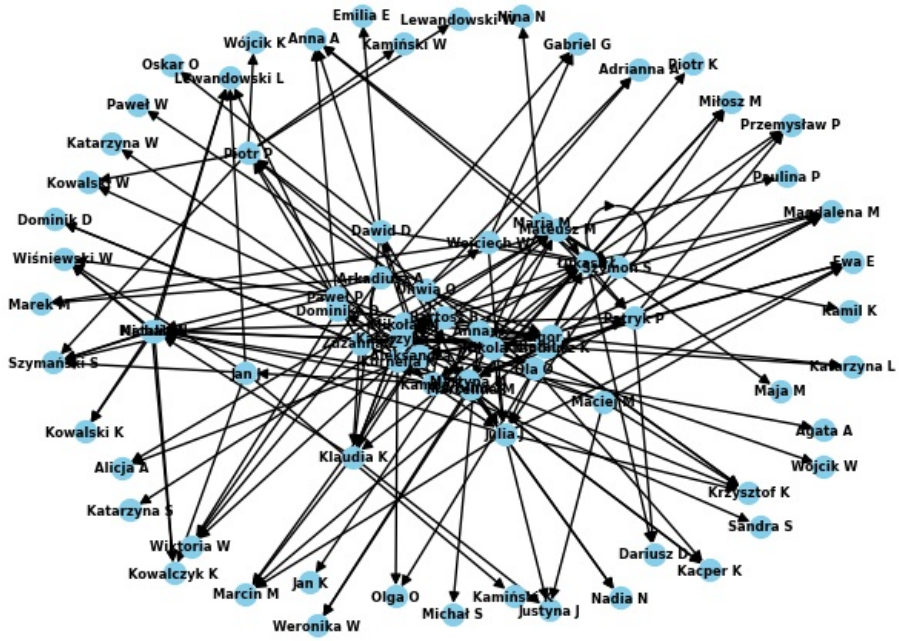


In [56]:

```
import networkx as nx
import matplotlib.pyplot as plt

# Tworzenie skierowanego grafu
G = nx.DiGraph()

# Dodawanie krawędzi z wcześniejszych połączeń
edges = [
    ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
```

```
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
    ("Wojciech W", "Kamil K"),
    ("Wojciech W", "Adrianna A"),
    ("Wojciech W", "Marek M"),
    ("Wojciech W", "Gabriel G"),
    ("Wojciech W", "Julia J"),
    ("Karolina K", "Dominik D"),
    ("Karolina K", "Weronika W"),
    ("Karolina K", "Miłosz M"),
    ("Karolina K", "Nadia N"),
    ("Karolina K", "Kacper K"),
    ("Bartosz B", "Magdalena M"),
    ("Bartosz B", "Mikołaj M"),
    ("Bartosz B", "Wiktoria W"),
    ("Bartosz B", "Piotr P"),
    ("Bartosz B", "Katarzyna K"),
    ("Kamila K", "Jan J"),
    ("Kamila K", "Natalia N"),
    ("Kamila K", "Bartosz B"),
    ("Kamila K", "Sandra S"),
    ("Kamila K", "Dawid D"),
    ("Dawid D", "Emilia E"),
    ("Dawid D", "Szymon S"),
    ("Dawid D", "Karolina K"),
    ("Dawid D", "Oskar O"),
    ("Dawid D", "Klaudia K"),
    ("Dominika D", "Paweł P"),
    ("Dominika D", "Wiktoria W"),
    ("Dominika D", "Igor I"),
    ("Dominika D", "Anna A"),
    ("Dominika D", "Kamila K"),
    ("Patryk P", "Dariusz D"),
    ("Patryk P", "Magdalena M"),
    ("Patryk P", "Przemysław P"),
    ("Patryk P", "Aleksandra A"),
    ("Patryk P", "Mikołaj M"),
    ("Julia J", "Justyna J"),
    ("Julia J", "Marcin M"),
    ("Julia J", "Natalia N"),
    ("Julia J", "Krzysztof K"),
    ("Julia J", "Ewa E"),
    ("Mikołaj M", "Łukasz Ł"),
    ("Mikołaj M", "Martyna M"),
    ("Mikołaj M", "Michał M"),
    ("Mikołaj M", "Klaudia K"),
    ("Mikołaj M", "Mateusz M"),
    ("Ola O", "Julia J"),
    ("Ola O", "Piotr P"),
    ("Ola O", "Olga O"),
    ("Ola O", "Kacper K"),
    ("Ola O", "Magdalena M"),
    ("Maciej M", "Mikołaj M"),
```

```
    ("Maciej M", "Justyna J"),
    ("Maciej M", "Katarzyna K"),
    ("Maciej M", "Krzysztof K"),
    ("Maciej M", "Ewa E"),
    ("Nikola N", "Łukasz Ł"),
    ("Nikola N", "Klaudia K"),
    ("Nikola N", "Mateusz M"),
    ("Nikola N", "Julia J"),
    ("Nikola N", "Ola O"),
    ("Zuzanna Z", "Klaudia K"),
    ("Zuzanna Z", "Mateusz M"),
    ("Zuzanna Z", "Piotr P"),
    ("Zuzanna Z", "Wiktoria W"),
    ("Zuzanna Z", "Igor I"),
    ("Łukasz Ł", "Anna A"),
    ("Łukasz Ł", "Kamila K"),
    ("Łukasz Ł", "Dariusz D"),
    ("Łukasz Ł", "Magdalena M"),
    ("Łukasz Ł", "Przemysław P"),
    ("Klaudia K", "Aleksandra A"),
    ("Klaudia K", "Mikołaj M"),
    ("Klaudia K", "Justyna J"),
    ("Klaudia K", "Marcin M"),
    ("Klaudia K", "Natalia N"),
    ("Rafał R", "Krzysztof K"),
    ("Rafał R", "Ewa E"),
    ("Rafał R", "Łukasz Ł"),
    ("Rafał R", "Martyna M"),
    ("Rafał R", "Michał M"),
    ("Kornelia K", "Klaudia K"),
    ("Kornelia K", "Mateusz M"),
    ("Kornelia K", "Julia J"),
    ("Kornelia K", "Piotr P"),
    ("Kornelia K", "Olga O"),
    ("Arkadiusz A", "Natalia N"),
    ("Arkadiusz A", "Paulina P"),
    ("Arkadiusz A", "Wiktoria W"),
    ("Arkadiusz A", "Marek M"),
    ("Arkadiusz A", "Gabriel G"),
    ("Marcelina M", "Julia J"),
    ("Marcelina M", "Dominik D"),
    ("Marcelina M", "Weronika W"),
    ("Marcelina M", "Miłosz M"),
    ("Marcelina M", "Nadia N"),
    ("Marcelina M", "Kacper K"),
    ("Igor I", "Magdalena M"),
    ("Igor I", "Przemysław P"),
    ("Igor I", "Aleksandra A"),
    ("Igor I", "Mikołaj M"),
    ("Igor I", "Marcin M"),
    ("Klaudiusz K", "Natalia N"),
    ("Klaudiusz K", "Krzysztof K"),
    ("Klaudiusz K", "Ewa E"),
    ("Klaudiusz K", "Łukasz Ł"),
    ("Klaudiusz K", "Martyna M"),
    ("Martyna M", "Dominika D"),
    ("Martyna M", "Patryk P"),
    ("Martyna M", "Julia J"),
    ("Martyna M", "Natalia N"),
    ("Martyna M", "Krzysztof K"),
]

G.add_edges_from(edges)

# Rysowanie grafu
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, font_weight='bold', node_size=100, node_color='skyblue', font_color='black',
plt.figure(figsize = (50,50))
# Wyświetlanie grafu
plt.show()
```

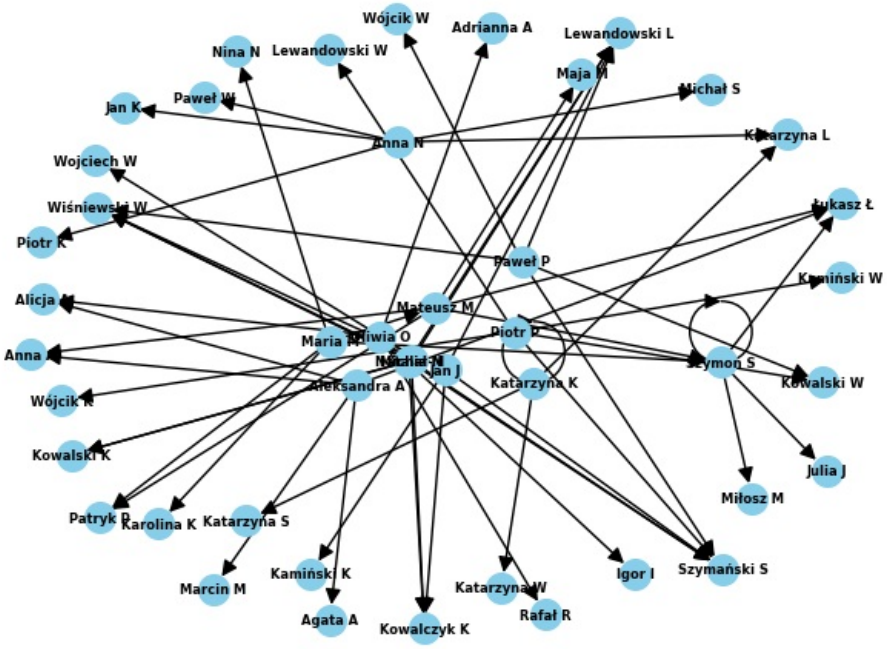


<Figure size 5000x5000 with 0 Axes>

In [92]:

```
import networkx as nx
import matplotlib.pyplot as plt

# Tworzenie skierowanego grafu
G = nx.DiGraph()

# Dodawanie krawędzi z wcześniejszych połączeń
edges = [
    ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
```

```
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),

]



G.add_edges_from(edges)

# Dodawanie pojemności 1 dla każdej krawędzi
for edge in G.edges:
    G[edge[0]][edge[1]]['capacity'] = 1

# Rysowanie grafu z większymi wierzchołkami
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, font_weight='bold', node_size=200, node_color='skyblue', font_color='black',

# Wyświetlanie grafu
plt.show()
```



In [63]:

```
import networkx as nx
import matplotlib.pyplot as plt

# Tworzenie skierowanego grafu
G = nx.DiGraph()

# Dodawanie krawędzi z wcześniejszych połączeń
edges = [
    ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
```

```
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
    ("Wojciech W", "Kamil K"),
    ("Wojciech W", "Adrianna A"),
    ("Wojciech W", "Marek M"),
    ("Wojciech W", "Gabriel G"),
    ("Wojciech W", "Julia J"),
    ("Karolina K", "Dominik D"),
    ("Karolina K", "Weronika W"),
    ("Karolina K", "Miłosz M"),
    ("Karolina K", "Nadia N"),
    ("Karolina K", "Kacper K"),
    ("Bartosz B", "Magdalena M"),
    ("Bartosz B", "Mikołaj M"),
    ("Bartosz B", "Wiktoria W"),
    ("Bartosz B", "Piotr P"),
    ("Bartosz B", "Katarzyna K"),
    ("Kamila K", "Jan J"),
    ("Kamila K", "Natalia N"),
    ("Kamila K", "Bartosz B"),
    ("Kamila K", "Sandra S"),
    ("Kamila K", "Dawid D"),
    ("Dawid D", "Emilia E"),
    ("Dawid D", "Szymon S"),
    ("Dawid D", "Karolina K"),
    ("Dawid D", "Oskar O"),
    ("Dawid D", "Klaudia K"),
    ("Dominika D", "Paweł P"),
    ("Dominika D", "Wiktoria W"),
    ("Dominika D", "Igor I"),
    ("Dominika D", "Anna A"),
    ("Dominika D", "Kamila K"),
    ("Patryk P", "Dariusz D"),
    ("Patryk P", "Magdalena M"),
    ("Patryk P", "Przemysław P"),
    ("Patryk P", "Aleksandra A"),
    ("Patryk P", "Mikołaj M"),
    ("Julia J", "Justyna J"),
    ("Julia J", "Marcin M"),
    ("Julia J", "Natalia N"),
    ("Julia J", "Krzysztof K"),
    ("Julia J", "Ewa E"),
    ("Mikołaj M", "Łukasz Ł"),
    ("Mikołaj M", "Martyna M"),
    ("Mikołaj M", "Michał M"),
    ("Mikołaj M", "Klaudia K"),
    ("Mikołaj M", "Mateusz M"),
    ("Ola O", "Julia J"),
```

```
    ("Ola O", "Piotr P"),
    ("Ola O", "Olga O"),
    ("Ola O", "Kacper K"),
    ("Ola O", "Magdalena M"),
    ("Maciej M", "Mikołaj M"),
    ("Maciej M", "Justyna J"),
    ("Maciej M", "Katarzyna K"),
    ("Maciej M", "Krzysztof K"),
    ("Maciej M", "Ewa E"),
    ("Nikola N", "Łukasz Ł"),
    ("Nikola N", "Klaudia K"),
    ("Nikola N", "Mateusz M"),
    ("Nikola N", "Julia J"),
    ("Nikola N", "Ola O"),
    ("Zuzanna Z", "Klaudia K"),
    ("Zuzanna Z", "Mateusz M"),
    ("Zuzanna Z", "Piotr P"),
    ("Zuzanna Z", "Wiktoria W"),
    ("Zuzanna Z", "Igor I"),
    ("Łukasz Ł", "Anna A"),
    ("Łukasz Ł", "Kamila K"),
    ("Łukasz Ł", "Dariusz D"),
    ("Łukasz Ł", "Magdalena M"),
    ("Łukasz Ł", "Przemysław P"),
    ("Klaudia K", "Aleksandra A"),
    ("Klaudia K", "Mikołaj M"),
    ("Klaudia K", "Justyna J"),
    ("Klaudia K", "Marcin M"),
    ("Klaudia K", "Natalia N"),
    ("Rafał R", "Krzysztof K"),
    ("Rafał R", "Ewa E"),
    ("Rafał R", "Łukasz Ł"),
    ("Rafał R", "Martyna M"),
    ("Rafał R", "Michał M"),
    ("Kornelia K", "Klaudia K"),
    ("Kornelia K", "Mateusz M"),
    ("Kornelia K", "Julia J"),
    ("Kornelia K", "Piotr P"),
    ("Kornelia K", "Olga O"),
    ("Arkadiusz A", "Natalia N"),
    ("Arkadiusz A", "Paulina P"),
    ("Arkadiusz A", "Wiktoria W"),
    ("Arkadiusz A", "Marek M"),
    ("Arkadiusz A", "Gabriel G"),
    ("Marcelina M", "Julia J"),
    ("Marcelina M", "Dominik D"),
    ("Marcelina M", "Weronika W"),
    ("Marcelina M", "Miłosz M"),
    ("Marcelina M", "Nadia N"),
    ("Marcelina M", "Kacper K"),
    ("Igor I", "Magdalena M"),
    ("Igor I", "Przemysław P"),
    ("Igor I", "Aleksandra A"),
    ("Igor I", "Mikołaj M"),
    ("Igor I", "Marcin M"),
    ("Klaudiusz K", "Natalia N"),
    ("Klaudiusz K", "Krzysztof K"),
    ("Klaudiusz K", "Ewa E"),
    ("Klaudiusz K", "Łukasz Ł"),
    ("Klaudiusz K", "Martyna M"),
    ("Martyna M", "Dominika D"),
    ("Martyna M", "Patryk P"),
    ("Martyna M", "Julia J"),
    ("Martyna M", "Natalia N"),
    ("Martyna M", "Krzysztof K"),
]

G.add_edges_from(edges)

# Dodawanie pojemności 1 dla każdej krawędzi
for edge in G.edges:
    G[edge[0]][edge[1]]['capacity'] = 1

# Obliczanie maksymalnego przepływu
source = "Anna N"
sink = "Natalia N"
max_flow_value, max_flow_dict = nx.maximum_flow(G, source, sink)

print("Maksymalny przepływ:", max_flow_value)
print("Przepływ przez krawędzie:", max_flow_dict)
```

```
Maksymalny przepływ: 0
Przepływ przez krawędzie: {'Anna N': {'Katarzyna L': 0, 'Piotr K': 0, 'Paweł W': 0, 'Michał S': 0, 'Jan K': 0},
'Katarzyna L': {}, 'Piotr K': {}, 'Paweł W': {}, 'Michał S': {}, 'Jan K': {}, 'Katarzyna K': {'Katarzyna W': 0,
'Katarzyna L': 0, 'Katarzyna K': 0, 'Katarzyna S': 0}, 'Katarzyna W': {}, 'Katarzyna S': {}, 'Piotr P': {'Lewan
dowski W': 0, 'Kowalski W': 0, 'Kamiński W': 0, 'Szymański S': 0, 'Wójcik K': 0}, 'Lewandowski W': {}, 'Kowalsk
i W': {}, 'Kamiński W': {}, 'Szymański S': {}, 'Wójcik K': {}, 'Paweł P': {'Wiśniewski W': 0, 'Kowalski W': 0,
'Wójcik W': 0, 'Szymański S': 0, 'Lewandowski L': 0}, 'Wiśniewski W': {}, 'Wójcik W': {}, 'Lewandowski L': {},
'Michał M': {'Wiśniewski W': 0, 'Lewandowski L': 0, 'Kowalczyk K': 0, 'Szymański S': 0, 'Kowalski K': 0}, 'Kowa
lczyk K': {}, 'Kowalski K': {}, 'Jan J': {'Lewandowski L': 0, 'Wiśniewski W': 0, 'Kamiński K': 0, 'Szymański S'
: 0, 'Kowalczyk K': 0}, 'Kamiński K': {}, 'Natalia N': {'Lewandowski L': 0, 'Kowalski K': 0, 'Wiśniewski W': 0,
'Szymański S': 0, 'Kowalczyk K': 0}, 'Mateusz M': {'Łukasz Ł': 0, 'Szymon S': 0, 'Patryk P': 0, 'Anna A': 0, 'M
aja M': 0}, 'Łukasz Ł': {'Anna A': 0, 'Kamila K': 0, 'Dariusz D': 0, 'Magdalena M': 0, 'Przemysław P': 0}, 'Szy
mon S': {'Julia J': 0, 'Szymon S': 0, 'Miłosz M': 0, 'Łukasz Ł': 0}, 'Patryk P': {'Dariusz D': 0, 'Magdalena M'
: 0, 'Przemysław P': 0, 'Aleksandra A': 0, 'Mikołaj M': 0}, 'Anna A': {}, 'Maja M': {}, 'Aleksandra A': {'Łukas
z Ł': 0, 'Anna A': 0, 'Agata A': 0, 'Alicja A': 0, 'Marcin M': 0}, 'Agata A': {}, 'Alicja A': {}, 'Marcin M': {
}, 'Maria M': {'Karolina K': 0, 'Mateusz M': 0, 'Szymon S': 0, 'Nina N': 0, 'Patryk P': 0}, 'Karolina K': {'Dom
inik D': 0, 'Weronika W': 0, 'Miłosz M': 0, 'Nadia N': 0, 'Kacper K': 0}, 'Nina N': {}, 'Julia J': {'Justyna J'
: 0, 'Marcin M': 0, 'Natalia N': 0, 'Krzysztof K': 0, 'Ewa E': 0}, 'Miłosz M': {}, 'Oliwia O': {'Adrianna A': 0
, 'Wojciech W': 0, 'Igor I': 0, 'Rafał R': 0, 'Alicja A': 0}, 'Adrianna A': {}, 'Wojciech W': {'Kamil K': 0, 'A
drianna A': 0, 'Marek M': 0, 'Gabriel G': 0, 'Julia J': 0}, 'Igor I': {'Magdalena M': 0, 'Przemysław P': 0, 'Al
eksandra A': 0, 'Mikołaj M': 0, 'Marcin M': 0}, 'Rafał R': {'Krzysztof K': 0, 'Ewa E': 0, 'Łukasz Ł': 0, 'Marty
na M': 0, 'Michał M': 0}, 'Kamil K': {}, 'Marek M': {}, 'Gabriel G': {}, 'Dominik D': {}, 'Weronika W': {}, 'Na
dia N': {}, 'Kacper K': {}, 'Bartosz B': {'Magdalena M': 0, 'Mikołaj M': 0, 'Wiktoria W': 0, 'Piotr P': 0, 'Kat
arzyna K': 0}, 'Magdalena M': {}, 'Mikołaj M': {'Łukasz Ł': 0, 'Martyna M': 0, 'Michał M': 0, 'Klaudia K': 0, '
Mateusz M': 0}, 'Wiktoria W': {}, 'Kamila K': {'Jan J': 0, 'Natalia N': 0, 'Bartosz B': 0, 'Sandra S': 0, 'Dawi
d D': 0}, 'Sandra S': {}, 'Dawid D': {'Emilia E': 0, 'Szymon S': 0, 'Karolina K': 0, 'Oskar O': 0, 'Klaudia K':
0}, 'Emilia E': {}, 'Oskar O': {}, 'Klaudia K': {'Aleksandra A': 0, 'Mikołaj M': 0, 'Justyna J': 0, 'Marcin M':
0, 'Natalia N': 0}, 'Dominika D': {'Paweł P': 0, 'Wiktoria W': 0, 'Igor I': 0, 'Anna A': 0, 'Kamila K': 0}, 'Da
riusz D': {}, 'Przemysław P': {}, 'Justyna J': {}, 'Krzysztof K': {}, 'Ewa E': {}, 'Martyna M': {'Dominika D':
0, 'Patryk P': 0, 'Julia J': 0, 'Natalia N': 0, 'Krzysztof K': 0}, 'Ola O': {'Julia J': 0, 'Piotr P': 0, 'Olga
O': 0, 'Kacper K': 0, 'Magdalena M': 0}, 'Olga O': {}, 'Maciej M': {'Mikołaj M': 0, 'Justyna J': 0, 'Katarzyna
K': 0, 'Krzysztof K': 0, 'Ewa E': 0}, 'Nikola N': {'Łukasz Ł': 0, 'Klaudia K': 0, 'Mateusz M': 0, 'Julia J': 0,
'Ola O': 0}, 'Zuzanna Z': {'Klaudia K': 0, 'Mateusz M': 0, 'Piotr P': 0, 'Wiktoria W': 0, 'Igor I': 0}, 'Kornel
ia K': {'Klaudia K': 0, 'Mateusz M': 0, 'Julia J': 0, 'Piotr P': 0, 'Olga O': 0}, 'Arkadiusz A': {'Natalia N':
0, 'Paulina P': 0, 'Wiktoria W': 0, 'Marek M': 0, 'Gabriel G': 0}, 'Paulina P': {}, 'Marcelina M': {'Julia J':
0, 'Dominik D': 0, 'Weronika W': 0, 'Miłosz M': 0, 'Nadia N': 0, 'Kacper K': 0}, 'Klaudiusz K': {'Natalia N': 0
, 'Krzysztof K': 0, 'Ewa E': 0, 'Łukasz Ł': 0, 'Martyna M': 0}}
```

In [66]:
```python
import math
from collections import Counter

def calculate_entropy(data):
    # Oblicz częstości występowania poszczególnych elementów w zbiorze danych
    data_count = Counter(data)

    # Oblicz prawdopodobieństwa dla każdego elementu
    probabilities = [count / len(data) for count in data_count.values()]

    # Oblicz entropię na podstawie wzoru Shannon entropy: H(X) = - Σ p(x) * log2(p(x))
    entropy = -sum(p * math.log2(p) for p in probabilities if p > 0)

    return entropy

# Przykładowe dane
sample_data = [1, 1, 2, 3, 3, 3, 4, 4, 4, 4]

# Oblicz entropię dla danych
entropy_value = calculate_entropy(sample_data)

print("Entropia:", entropy_value)
```

```
Entropia: 1.8464393446710154
```

In [67]:
```python
import math
from collections import Counter

# Dane z poprzednich połączeń
data = [
    "Anna N", "Katarzyna L", "Anna N", "Piotr K", "Anna N",
    "Paweł W", "Anna N", "Michał S", "Anna N", "Jan K",
    "Katarzyna K", "Katarzyna W", "Katarzyna K", "Katarzyna L", "Katarzyna K",
    "Piotr P", "Lewandowski W", "Piotr P", "Kowalski W", "Piotr P",
    "Kamiński W", "Piotr P", "Szymański S", "Piotr P", "Wójcik K",
    "Paweł P", "Wiśniewski W", "Paweł P", "Kowalski W", "Paweł P",
    "Wójcik W", "Paweł P", "Szymański S", "Paweł P", "Lewandowski L",
    "Michał M", "Wiśniewski W", "Michał M", "Lewandowski L", "Michał M",
    "Kowalczyk K", "Michał M", "Szymański S", "Michał M", "Kowalski K",
    "Jan J", "Lewandowski L", "Jan J", "Wiśniewski W", "Jan J",
    "Kamiński K", "Jan J", "Szymański S", "Jan J", "Kowalczyk K",
    "Natalia N", "Lewandowski L", "Natalia N", "Kowalski K", "Natalia N",
    "Wiśniewski W", "Natalia N", "Szymański S", "Natalia N", "Kowalczyk K",
    "Mateusz M", "Łukasz Ł", "Mateusz M", "Szymon S", "Mateusz M",
    "Patryk P", "Mateusz M", "Anna A", "Mateusz M", "Maja M",
    "Aleksandra A", "Łukasz Ł", "Aleksandra A", "Anna A", "Aleksandra A",
    "Agata A", "Aleksandra A", "Alicja A", "Aleksandra A", "Marcin M",
    "Maria M", "Karolina K", "Maria M", "Mateusz M", "Maria M",
    "Szymon S", "Maria M", "Nina N", "Maria M", "Patryk P",
    "Szymon S", "Julia J", "Szymon S", "Szymon S", "Szymon S",
    "Miłosz M", "Szymon S", "Łukasz Ł", "Szymon S", "Łukasz Ł",
```

```
    "Oliwia O", "Adrianna A", "Oliwia O", "Wojciech W", "Oliwia O",
    "Igor I", "Oliwia O", "Rafał R", "Oliwia O", "Alicja A",
    "Wojciech W", "Kamil K", "Wojciech W", "Adrianna A", "Wojciech W",
    "Marek M", "Wojciech W", "Gabriel G", "Wojciech W", "Julia J",
    "Karolina K", "Dominik D", "Karolina K", "Weronika W", "Karolina K",
    "Miłosz M", "Karolina K", "Nadia N", "Karolina K", "Kacper K",
    "Bartosz B", "Magdalena M", "Bartosz B", "Mikołaj M", "Bartosz B",
    "Wiktoria W", "Bartosz B", "Piotr P", "Bartosz B", "Katarzyna K",
    "Kamila K", "Jan J", "Kamila K", "Natalia N", "Kamila K",
    "Bartosz B", "Kamila K", "Sandra S", "Kamila K", "Dawid D",
    "Dawid D", "Emilia E", "Dawid D", "Szymon S", "Dawid D",
    "Karolina K", "Dawid D", "Oskar O", "Dawid D", "Klaudia K",
    "Dominika D", "Paweł P", "Dominika D", "Wiktoria W", "Dominika D",
    "Igor I", "Dominika D", "Anna A", "Dominika D", "Kamila K",
    "Patryk P", "Dariusz D", "Patryk P", "Magdalena M", "Patryk P",
    "Przemysław P", "Patryk P", "Aleksandra A", "Patryk P", "Mikołaj M",
    "Julia J", "Justyna J", "Julia J", "Marcin M", "Julia J",
    "Natalia N", "Julia J", "Krzysztof K", "Julia J", "Ewa E",
    "Mikołaj M", "Łukasz Ł", "Mikołaj M", "Martyna M", "Mikołaj M",
    "Michał M", "Mikołaj M", "Klaudia K", "Mikołaj M", "Mateusz M",
    "Ola O", "Julia J", "Ola O", "Piotr P", "Ola O",
    "Olga O", "Ola O", "Kacper K", "Ola O", "Magdalena M",
    "Maciej M", "Mikołaj M", "Maciej M", "Justyna J", "Maciej M",
    "Katarzyna K", "Maciej M", "Krzysztof K", "Maciej M", "Ewa E",
    "Nikola N", "Łukasz Ł", "Nikola N", "Klaudia K", "Nikola N",
    "Mateusz M", "Nikola N", "Julia J", "Nikola N", "Ola O",
    "Zuzanna Z", "Klaudia K", "Zuzanna Z", "Mateusz M", "Zuzanna Z",
    "Piotr P", "Zuzanna Z", "Wiktoria W", "Zuzanna Z", "Igor I",
    "Łukasz Ł", "Anna A", "Łukasz Ł", "Kamila K", "Łukasz Ł",
    "Dariusz D", "Łukasz Ł", "Magdalena M", "Łukasz Ł", "Przemysław P",
    "Klaudia K", "Aleksandra A", "Klaudia K", "Mikołaj M", "Klaudia K",
    "Justyna J", "Klaudia K", "Marcin M", "Klaudia K", "Natalia N",
    "Rafał R", "Krzysztof K", "Rafał R", "Ewa E", "Rafał R",
    "Łukasz Ł", "Rafał R", "Martyna M", "Rafał R", "Michał M",
    "Kornelia K", "Klaudia K", "Kornelia K", "Mateusz M", "Kornelia K",
    "Julia J", "Kornelia K", "Piotr P", "Kornelia K", "Olga O",
    "Arkadiusz A", "Natalia N", "Arkadiusz A", "Paulina P", "Arkadiusz A",
    "Wiktoria W", "Arkadiusz A", "Marek M", "Arkadiusz A", "Gabriel G",
    "Marcelina M", "Julia J", "Marcelina M", "Dominik D", "Marcelina M",
    "Weronika W", "Marcelina M", "Miłosz M", "Marcelina M", "Nadia N",
    "Marcelina M", "Kacper K", "Igor I", "Magdalena M", "Igor I",
    "Przemysław P", "Igor I", "Aleksandra A", "Igor I", "Mikołaj M",
    "Igor I", "Marcin M", "Klaudiusz K", "Natalia N", "Klaudiusz K",
    "Krzysztof K", "Klaudiusz K", "Ewa E", "Klaudiusz K", "Łukasz Ł",
    "Klaudiusz K", "Martyna M", "Martyna M", "Dominika D", "Martyna M",
    "Patryk P", "Martyna M", "Julia J", "Martyna M", "Natalia N",
    "Martyna M", "Krzysztof K"
]

# Oblicz entropię dla danych
entropy_value = calculate_entropy(data)

print("Entropia:", entropy_value)
```

```
Entropia: 5.942190243785034
```

In [69]:

```
import networkx as nx
import matplotlib.pyplot as plt

# Tworzenie skierowanego grafu
G = nx.DiGraph()

# Dodawanie krawędzi z wcześniejszych połączeń
edges = [
   ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
```

```
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
    ("Wojciech W", "Kamil K"),
    ("Wojciech W", "Adrianna A"),
    ("Wojciech W", "Marek M"),
    ("Wojciech W", "Gabriel G"),
    ("Wojciech W", "Julia J"),
    ("Karolina K", "Dominik D"),
    ("Karolina K", "Weronika W"),
    ("Karolina K", "Miłosz M"),
    ("Karolina K", "Nadia N"),
    ("Karolina K", "Kacper K"),
    ("Bartosz B", "Magdalena M"),
    ("Bartosz B", "Mikołaj M"),
    ("Bartosz B", "Wiktoria W"),
    ("Bartosz B", "Piotr P"),
    ("Bartosz B", "Katarzyna K"),
    ("Kamila K", "Jan J"),
    ("Kamila K", "Natalia N"),
    ("Kamila K", "Bartosz B"),
    ("Kamila K", "Sandra S"),
    ("Kamila K", "Dawid D"),
    ("Dawid D", "Emilia E"),
    ("Dawid D", "Szymon S"),
    ("Dawid D", "Karolina K"),
    ("Dawid D", "Oskar O"),
    ("Dawid D", "Klaudia K"),
    ("Dominika D", "Paweł P"),
    ("Dominika D", "Wiktoria W"),
    ("Dominika D", "Igor I"),
    ("Dominika D", "Anna A"),
    ("Dominika D", "Kamila K"),
    ("Patryk P", "Dariusz D"),
    ("Patryk P", "Magdalena M"),
    ("Patryk P", "Przemysław P"),
    ("Patryk P", "Aleksandra A"),
    ("Patryk P", "Mikołaj M"),
    ("Julia J", "Justyna J"),
    ("Julia J", "Marcin M"),
    ("Julia J", "Natalia N"),
    ("Julia J", "Krzysztof K"),
    ("Julia J", "Ewa E"),
    ("Mikołaj M", "Łukasz Ł"),
    ("Mikołaj M", "Martyna M"),
    ("Mikołaj M", "Michał M"),
    ("Mikołaj M", "Klaudia K"),
    ("Mikołaj M", "Mateusz M"),
    ("Ola O", "Julia J"),
    ("Ola O", "Piotr P"),
    ("Ola O", "Olga O"),
    ("Ola O", "Kacper K"),
    ("Ola O", "Magdalena M"),
    ("Maciej M", "Mikołaj M"),
    ("Maciej M", "Justyna J"),
    ("Maciej M", "Katarzyna K"),
    ("Maciej M", "Krzysztof K"),
```

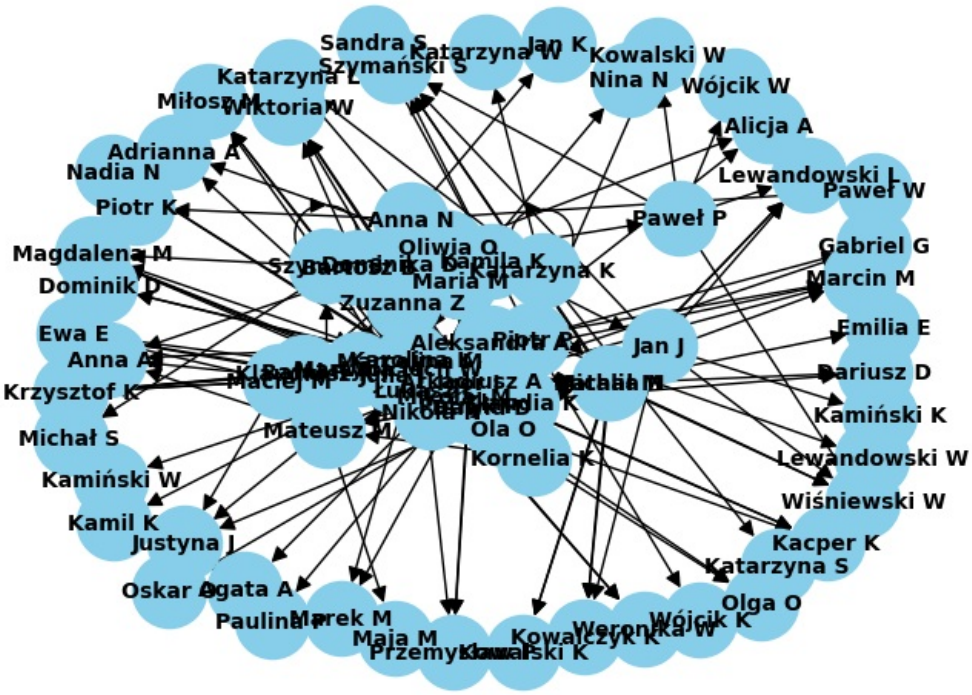
```
    ("Maciej M", "Ewa E"),
    ("Nikola N", "Łukasz Ł"),
    ("Nikola N", "Klaudia K"),
    ("Nikola N", "Mateusz M"),
    ("Nikola N", "Julia J"),
    ("Nikola N", "Ola O"),
    ("Zuzanna Z", "Klaudia K"),
    ("Zuzanna Z", "Mateusz M"),
    ("Zuzanna Z", "Piotr P"),
    ("Zuzanna Z", "Wiktoria W"),
    ("Zuzanna Z", "Igor I"),
    ("Łukasz Ł", "Anna A"),
    ("Łukasz Ł", "Kamila K"),
    ("Łukasz Ł", "Dariusz D"),
    ("Łukasz Ł", "Magdalena M"),
    ("Łukasz Ł", "Przemysław P"),
    ("Klaudia K", "Aleksandra A"),
    ("Klaudia K", "Mikołaj M"),
    ("Klaudia K", "Justyna J"),
    ("Klaudia K", "Marcin M"),
    ("Klaudia K", "Natalia N"),
    ("Rafał R", "Krzysztof K"),
    ("Rafał R", "Ewa E"),
    ("Rafał R", "Łukasz Ł"),
    ("Rafał R", "Martyna M"),
    ("Rafał R", "Michał M"),
    ("Kornelia K", "Klaudia K"),
    ("Kornelia K", "Mateusz M"),
    ("Kornelia K", "Julia J"),
    ("Kornelia K", "Piotr P"),
    ("Kornelia K", "Olga O"),
    ("Arkadiusz A", "Natalia N"),
    ("Arkadiusz A", "Paulina P"),
    ("Arkadiusz A", "Wiktoria W"),
    ("Arkadiusz A", "Marek M"),
    ("Arkadiusz A", "Gabriel G"),
    ("Marcelina M", "Julia J"),
    ("Marcelina M", "Dominik D"),
    ("Marcelina M", "Weronika W"),
    ("Marcelina M", "Miłosz M"),
    ("Marcelina M", "Nadia N"),
    ("Marcelina M", "Kacper K"),
    ("Igor I", "Magdalena M"),
    ("Igor I", "Przemysław P"),
    ("Igor I", "Aleksandra A"),
    ("Igor I", "Mikołaj M"),
    ("Igor I", "Marcin M"),
    ("Klaudiusz K", "Natalia N"),
    ("Klaudiusz K", "Krzysztof K"),
    ("Klaudiusz K", "Ewa E"),
    ("Klaudiusz K", "Łukasz Ł"),
    ("Klaudiusz K", "Martyna M"),
    ("Martyna M", "Dominika D"),
    ("Martyna M", "Patryk P"),
    ("Martyna M", "Julia J"),
    ("Martyna M", "Natalia N"),
    ("Martyna M", "Krzysztof K"),
]

G.add_edges_from(edges)

# Obliczanie centralności w stopniu
degree_centrality = nx.degree_centrality(G)

# Obliczanie centralności pośrednictwa
betweenness_centrality = nx.betweenness_centrality(G)

# Obliczanie centralności bliskości
closeness_centrality = nx.closeness_centrality(G)

# Wyświetlanie wyników
print("Centralność w stopniu:")
for node, centrality in degree_centrality.items():
    print(f"{node}: {centrality}")

print("\nCentralność pośrednictwa:")
for node, centrality in betweenness_centrality.items():
    print(f"{node}: {centrality}")

print("\nCentralność bliskości:")
for node, centrality in closeness_centrality.items():
    print(f"{node}: {centrality}")

# Rysowanie grafu
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, font_weight='bold', node_size=1200, node_color='skyblue', font_color='black',

# Wyświetlanie grafu
plt.show()
```

```
Centralność w stopniu:
Anna N: 0.0641025641025641
Katarzyna L: 0.02564102564102564
Piotr K: 0.01282051282051282
Paweł W: 0.01282051282051282
Michał S: 0.01282051282051282
Jan K: 0.01282051282051282
Katarzyna K: 0.08974358974358974
Katarzyna W: 0.01282051282051282
Katarzyna S: 0.01282051282051282
Piotr P: 0.11538461538461538
Lewandowski W: 0.01282051282051282
Kowalski W: 0.02564102564102564
Kamiński W: 0.01282051282051282
Szymański S: 0.0641025641025641
Wójcik K: 0.01282051282051282
Paweł P: 0.07692307692307693
Wiśniewski W: 0.05128205128205128
Wójcik W: 0.01282051282051282
Lewandowski L: 0.05128205128205128
Michał M: 0.08974358974358974
Kowalczyk K: 0.038461538461538464
Kowalski K: 0.02564102564102564
Jan J: 0.07692307692307693
Kamiński K: 0.01282051282051282
Natalia N: 0.14102564102564102
Mateusz M: 0.1282051282051282
Łukasz Ł: 0.15384615384615385
Szymon S: 0.10256410256410256
Patryk P: 0.10256410256410256
Anna A: 0.05128205128205128
Maja M: 0.01282051282051282
Aleksandra A: 0.10256410256410256
Agata A: 0.01282051282051282
Alicja A: 0.02564102564102564
Marcin M: 0.05128205128205128
Maria M: 0.0641025641025641
Karolina K: 0.08974358974358974
Nina N: 0.01282051282051282
Julia J: 0.15384615384615385
Miłosz M: 0.038461538461538464
Oliwia O: 0.0641025641025641
Adrianna A: 0.02564102564102564
Wojciech W: 0.07692307692307693
Igor I: 0.10256410256410256
Rafał R: 0.07692307692307693
Kamil K: 0.01282051282051282
Marek M: 0.02564102564102564
Gabriel G: 0.02564102564102564
Dominik D: 0.02564102564102564
Weronika W: 0.02564102564102564
Nadia N: 0.02564102564102564
Kacper K: 0.038461538461538464
Bartosz B: 0.07692307692307693
Magdalena M: 0.0641025641025641
Mikołaj M: 0.1282051282051282
Wiktoria W: 0.05128205128205128
Kamila K: 0.08974358974358974
Sandra S: 0.01282051282051282
Dawid D: 0.07692307692307693
Emilia E: 0.01282051282051282
Oskar O: 0.01282051282051282
Klaudia K: 0.1282051282051282
Dominika D: 0.07692307692307693
Dariusz D: 0.02564102564102564
Przemysław P: 0.038461538461538464
Justyna J: 0.038461538461538464
Krzysztof K: 0.0641025641025641
Ewa E: 0.05128205128205128
Martyna M: 0.10256410256410256
Ola O: 0.07692307692307693
Olga O: 0.02564102564102564
Maciej M: 0.0641025641025641
Nikola N: 0.0641025641025641
Zuzanna Z: 0.0641025641025641
Kornelia K: 0.0641025641025641
Arkadiusz A: 0.0641025641025641
Paulina P: 0.01282051282051282
Marcelina M: 0.07692307692307693
Klaudiusz K: 0.0641025641025641

Centralność pośrednictwa:
Anna N: 0.0
Katarzyna L: 0.0
Piotr K: 0.0
Paweł W: 0.0
Michał S: 0.0
Jan K: 0.0
```

Katarzyna K: 0.01048951048951049
Katarzyna W: 0.0
Katarzyna S: 0.0
Piotr P: 0.01356976356976357
Lewandowski W: 0.0
Kowalski W: 0.0
Kamiński W: 0.0
Szymański S: 0.0
Wójcik K: 0.0
Paweł P: 0.005799755799755799
Wiśniewski W: 0.0
Wójcik W: 0.0
Lewandowski L: 0.0
Michał M: 0.006299256299256298
Kowalczyk K: 0.0
Kowalski K: 0.0
Jan J: 0.004745254745254745
Kamiński K: 0.0
Natalia N: 0.012876012876012876
Mateusz M: 0.017746142746142744
Łukasz Ł: 0.06929181929181928
Szymon S: 0.012196137196137196
Patryk P: 0.010128760128760128
Anna A: 0.0
Maja M: 0.0
Aleksandra A: 0.013611388611388612
Agata A: 0.0
Alicja A: 0.0
Marcin M: 0.0
Maria M: 0.0
Karolina K: 0.01454101454101454
Nina N: 0.0
Julia J: 0.01415251415251415
Miłosz M: 0.0
Oliwia O: 0.0
Adrianna A: 0.0
Wojciech W: 0.0009157509157509158
Igor I: 0.005966255966255967
Rafał R: 0.006493506493506494
Kamil K: 0.0
Marek M: 0.0
Gabriel G: 0.0
Dominik D: 0.0
Weronika W: 0.0
Nadia N: 0.0
Kacper K: 0.0
Bartosz B: 0.03368853368853369
Magdalena M: 0.0
Mikołaj M: 0.049367299367299375
Wiktoria W: 0.0
Kamila K: 0.07539682539682539
Sandra S: 0.0
Dawid D: 0.02958152958152958
Emilia E: 0.0
Oskar O: 0.0
Klaudia K: 0.015886890886890884
Dominika D: 0.016705516705516708
Dariusz D: 0.0
Przemysław P: 0.0
Justyna J: 0.0
Krzysztof K: 0.0
Ewa E: 0.0
Martyna M: 0.023948273948273945
Ola O: 0.0013042513042513043
Olga O: 0.0
Maciej M: 0.0
Nikola N: 0.0
Zuzanna Z: 0.0
Kornelia K: 0.0
Arkadiusz A: 0.0
Paulina P: 0.0
Marcelina M: 0.0
Klaudiusz K: 0.0

Centralność bliskości:
Anna N: 0.0
Katarzyna L: 0.0645909645909646
Piotr K: 0.01282051282051282
Paweł W: 0.01282051282051282
Michał S: 0.01282051282051282
Jan K: 0.01282051282051282
Katarzyna K: 0.06894934333958724
Katarzyna W: 0.059664694280078895
Katarzyna S: 0.059664694280078895
Piotr P: 0.08058608058608058
Lewandowski W: 0.06782051282051282
Kowalski W: 0.08297320656871218
Kamiński W: 0.06782051282051282
Szymański S: 0.168774148226203

Wójcik K: 0.06782051282051282
Paweł P: 0.0621301775147929
Wiśniewski W: 0.15182186234817815
Wójcik W: 0.054912638983435444
Lewandowski L: 0.15182186234817815
Michał M: 0.09582790091264667
Kowalczyk K: 0.14186909581646426
Kowalski K: 0.13401709401709402
Jan J: 0.08193979933110368
Kamiński K: 0.06818822203437588
Natalia N: 0.16049382716049382
Mateusz M: 0.09861932938856015
Łukasz Ł: 0.14245014245014245
Szymon S: 0.09496676163342829
Patryk P: 0.08691873098652758
Anna A: 0.1284965034965035
Maja M: 0.07744994731296101
Aleksandra A: 0.10911074740861974
Agata A: 0.0831447963800905
Alicja A: 0.08566433566433566
Marcin M: 0.1456876456876457
Maria M: 0.0
Karolina K: 0.06894934333958724
Nina N: 0.01282051282051282
Julia J: 0.13426573426573427
Miłosz M: 0.08807858807858808
Oliwia O: 0.0
Adrianna A: 0.02564102564102564
Wojciech W: 0.01282051282051282
Igor I: 0.0641025641025641
Rafał R: 0.01282051282051282
Kamil K: 0.017094017094017092
Marek M: 0.028846153846153848
Gabriel G: 0.028846153846153848
Dominik D: 0.0645909645909646
Weronika W: 0.0645909645909646
Nadia N: 0.0645909645909646
Kacper K: 0.07169529499626588
Bartosz B: 0.07889546351084813
Magdalena M: 0.14430530709600478
Mikołaj M: 0.12210012210012208
Wiktoria W: 0.08739617190321415
Kamila K: 0.10683760683760683
Sandra S: 0.08193979933110368
Dawid D: 0.07889546351084813
Emilia E: 0.06574239713774598
Oskar O: 0.06574239713774598
Klaudia K: 0.10465724751439036
Dominika D: 0.07326007326007325
Dariusz D: 0.10872781065088757
Przemysław P: 0.1229096989966555
Justyna J: 0.1271876271876272
Krzysztof K: 0.12327416173570022
Ewa E: 0.1112891737891738
Martyna M: 0.09675858732462506
Ola O: 0.01282051282051282
Olga O: 0.028846153846153848
Maciej M: 0.0
Nikola N: 0.0
Zuzanna Z: 0.0
Kornelia K: 0.0
Arkadiusz A: 0.0
Paulina P: 0.01282051282051282
Marcelina M: 0.0
Klaudiusz K: 0.0



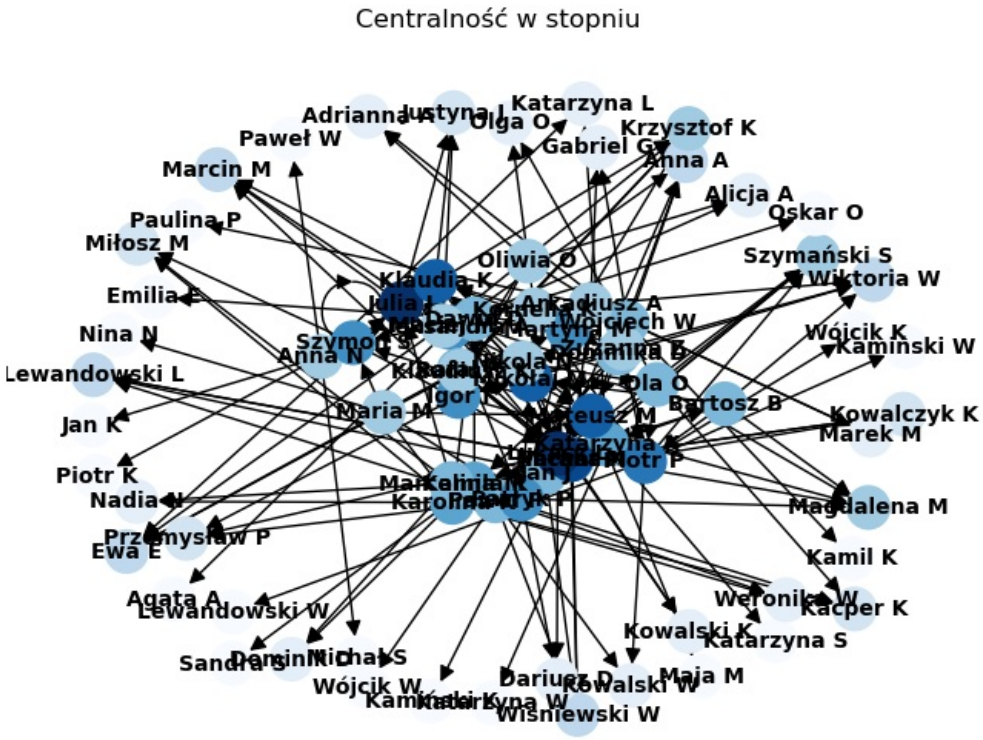
In [2]:

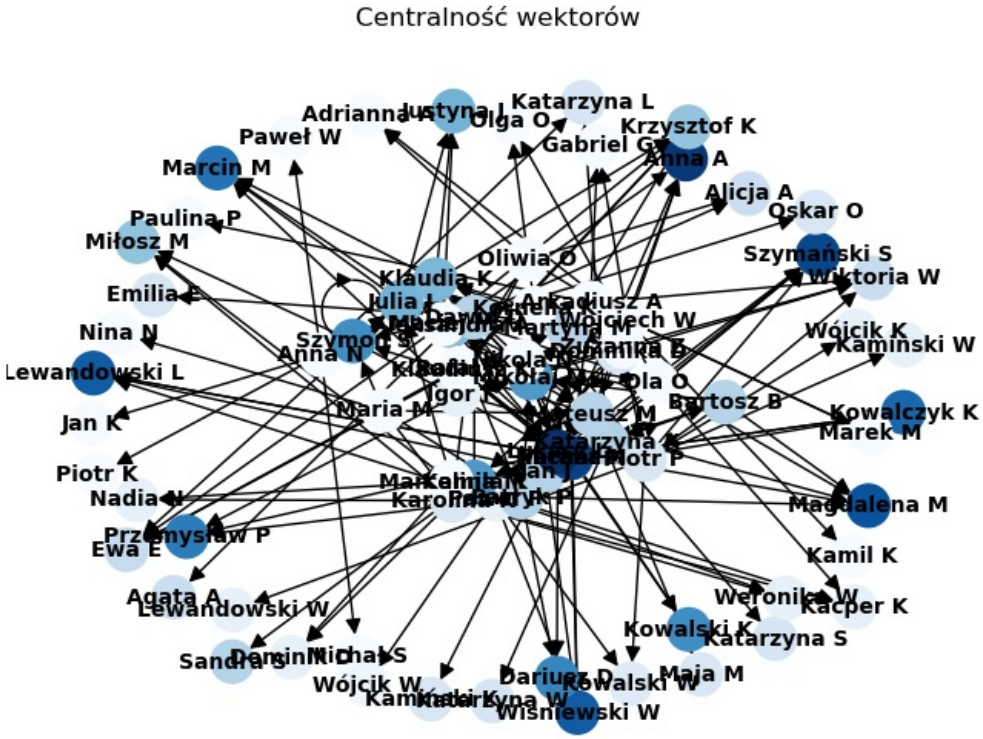
```
import networkx as nx
import matplotlib.pyplot as plt

# Tworzenie skierowanego grafu
G = nx.DiGraph()

# Dodawanie krawędzi z wcześniejszych połączeń
edges = [

   ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
```

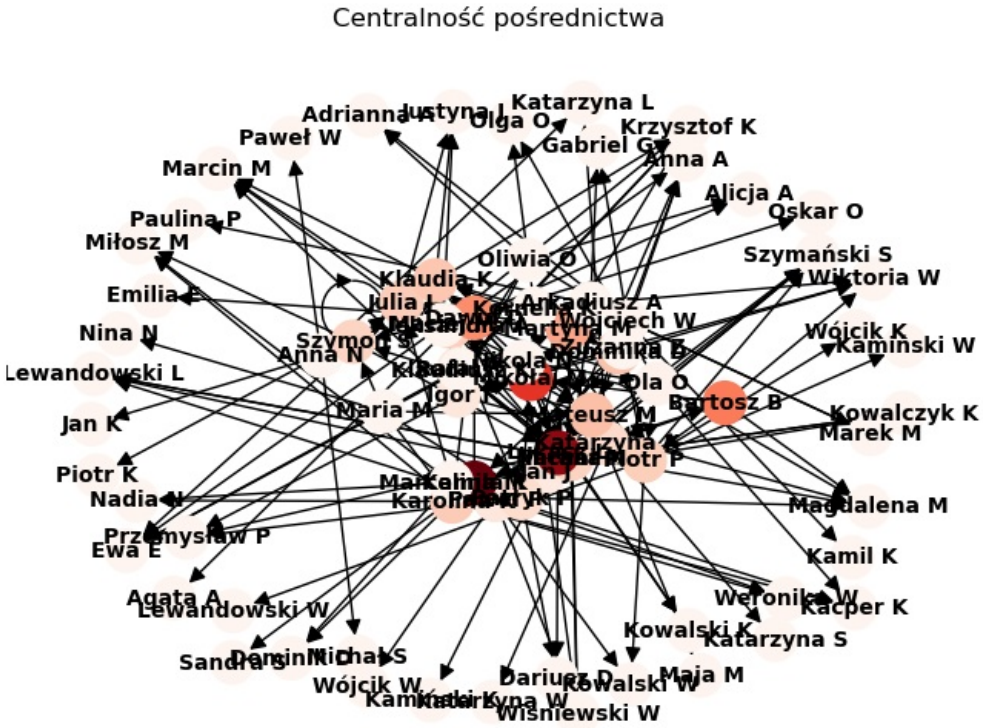
```
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
    ("Wojciech W", "Kamil K"),
    ("Wojciech W", "Adrianna A"),
    ("Wojciech W", "Marek M"),
    ("Wojciech W", "Gabriel G"),
    ("Wojciech W", "Julia J"),
    ("Karolina K", "Dominik D"),
    ("Karolina K", "Weronika W"),
    ("Karolina K", "Miłosz M"),
    ("Karolina K", "Nadia N"),
    ("Karolina K", "Kacper K"),
    ("Bartosz B", "Magdalena M"),
    ("Bartosz B", "Mikołaj M"),
    ("Bartosz B", "Wiktoria W"),
    ("Bartosz B", "Piotr P"),
    ("Bartosz B", "Katarzyna K"),
    ("Kamila K", "Jan J"),
    ("Kamila K", "Natalia N"),
    ("Kamila K", "Bartosz B"),
    ("Kamila K", "Sandra S"),
    ("Kamila K", "Dawid D"),
    ("Dawid D", "Emilia E"),
    ("Dawid D", "Szymon S"),
    ("Dawid D", "Karolina K"),
    ("Dawid D", "Oskar O"),
    ("Dawid D", "Klaudia K"),
    ("Dominika D", "Paweł P"),
    ("Dominika D", "Wiktoria W"),
    ("Dominika D", "Igor I"),
    ("Dominika D", "Anna A"),
    ("Dominika D", "Kamila K"),
    ("Patryk P", "Dariusz D"),
    ("Patryk P", "Magdalena M"),
    ("Patryk P", "Przemysław P"),
    ("Patryk P", "Aleksandra A"),
    ("Patryk P", "Mikołaj M"),
    ("Julia J", "Justyna J"),
    ("Julia J", "Marcin M"),
    ("Julia J", "Natalia N"),
    ("Julia J", "Krzysztof K"),
    ("Julia J", "Ewa E"),
    ("Mikołaj M", "Łukasz Ł"),
    ("Mikołaj M", "Martyna M"),
    ("Mikołaj M", "Michał M"),
    ("Mikołaj M", "Klaudia K"),
    ("Mikołaj M", "Mateusz M"),
    ("Ola O", "Julia J"),
    ("Ola O", "Piotr P"),
    ("Ola O", "Olga O"),
    ("Ola O", "Kacper K"),
    ("Ola O", "Magdalena M"),
    ("Maciej M", "Mikołaj M"),
    ("Maciej M", "Justyna J"),
```

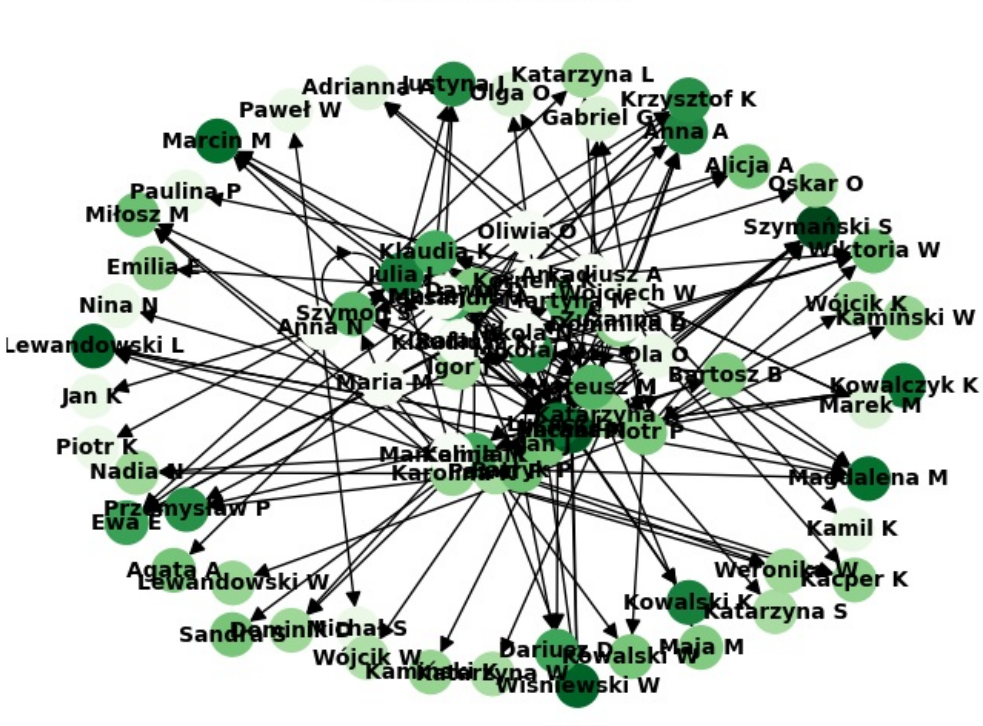
```
    ("Maciej M", "Katarzyna K"),
    ("Maciej M", "Krzysztof K"),
    ("Maciej M", "Ewa E"),
    ("Nikola N", "Łukasz Ł"),
    ("Nikola N", "Klaudia K"),
    ("Nikola N", "Mateusz M"),
    ("Nikola N", "Julia J"),
    ("Nikola N", "Ola O"),
    ("Zuzanna Z", "Klaudia K"),
    ("Zuzanna Z", "Mateusz M"),
    ("Zuzanna Z", "Piotr P"),
    ("Zuzanna Z", "Wiktoria W"),
    ("Zuzanna Z", "Igor I"),
    ("Łukasz Ł", "Anna A"),
    ("Łukasz Ł", "Kamila K"),
    ("Łukasz Ł", "Dariusz D"),
    ("Łukasz Ł", "Magdalena M"),
    ("Łukasz Ł", "Przemysław P"),
    ("Klaudia K", "Aleksandra A"),
    ("Klaudia K", "Mikołaj M"),
    ("Klaudia K", "Justyna J"),
    ("Klaudia K", "Marcin M"),
    ("Klaudia K", "Natalia N"),
    ("Rafał R", "Krzysztof K"),
    ("Rafał R", "Ewa E"),
    ("Rafał R", "Łukasz Ł"),
    ("Rafał R", "Martyna M"),
    ("Rafał R", "Michał M"),
    ("Kornelia K", "Klaudia K"),
    ("Kornelia K", "Mateusz M"),
    ("Kornelia K", "Julia J"),
    ("Kornelia K", "Piotr P"),
    ("Kornelia K", "Olga O"),
    ("Arkadiusz A", "Natalia N"),
    ("Arkadiusz A", "Paulina P"),
    ("Arkadiusz A", "Wiktoria W"),
    ("Arkadiusz A", "Marek M"),
    ("Arkadiusz A", "Gabriel G"),
    ("Marcelina M", "Julia J"),
    ("Marcelina M", "Dominik D"),
    ("Marcelina M", "Weronika W"),
    ("Marcelina M", "Miłosz M"),
    ("Marcelina M", "Nadia N"),
    ("Marcelina M", "Kacper K"),
    ("Igor I", "Magdalena M"),
    ("Igor I", "Przemysław P"),
    ("Igor I", "Aleksandra A"),
    ("Igor I", "Mikołaj M"),
    ("Igor I", "Marcin M"),
    ("Klaudiusz K", "Natalia N"),
    ("Klaudiusz K", "Krzysztof K"),
    ("Klaudiusz K", "Ewa E"),
    ("Klaudiusz K", "Łukasz Ł"),
    ("Klaudiusz K", "Martyna M"),
    ("Martyna M", "Dominika D"),
    ("Martyna M", "Patryk P"),
    ("Martyna M", "Julia J"),
    ("Martyna M", "Natalia N"),
    ("Martyna M", "Krzysztof K"),

]

G.add_edges_from(edges)

# Obliczanie centralności w stopniu
degree_centrality = nx.degree_centrality(G)

# Obliczanie centralności pośrednictwa
betweenness_centrality = nx.betweenness_centrality(G)

# Obliczanie centralności bliskości
closeness_centrality = nx.closeness_centrality(G)

# Obliczanie centralności bliskości
eigenvector_centrality = nx.eigenvector_centrality(G)

# Rysowanie grafu z oznaczeniem centralności kolorami
pos = nx.spring_layout(G)

# Centralność w stopniu
node_colors_degree = [degree_centrality[node] for node in G.nodes]
nx.draw(G, pos, with_labels=True, font_weight='bold', node_size=400, node_color=node_colors_degree, cmap=plt.cm
plt.title("Centralność w stopniu")
plt.show()

# Centralność wektorów
node_colors_degree = [eigenvector_centrality[node] for node in G.nodes]
nx.draw(G, pos, with_labels=True, font_weight='bold', node_size=400, node_color=node_colors_degree, cmap=plt.cm
plt.title("Centralność wektorów")
```

```
plt.show()

# Centralność pośrednictwa
node_colors_betweenness = [betweenness_centrality[node] for node in G.nodes]
nx.draw(G, pos, with_labels=True, font_weight='bold', node_size=400, node_color=node_colors_betweenness, cmap=p
plt.title("Centralność pośrednictwa")
plt.show()

# Centralność bliskości
node_colors_closeness = [closeness_centrality[node] for node in G.nodes]
nx.draw(G, pos, with_labels=True, font_weight='bold', node_size=400, node_color=node_colors_closeness, cmap=plt
plt.title("Centralność bliskości")
plt.show()
```









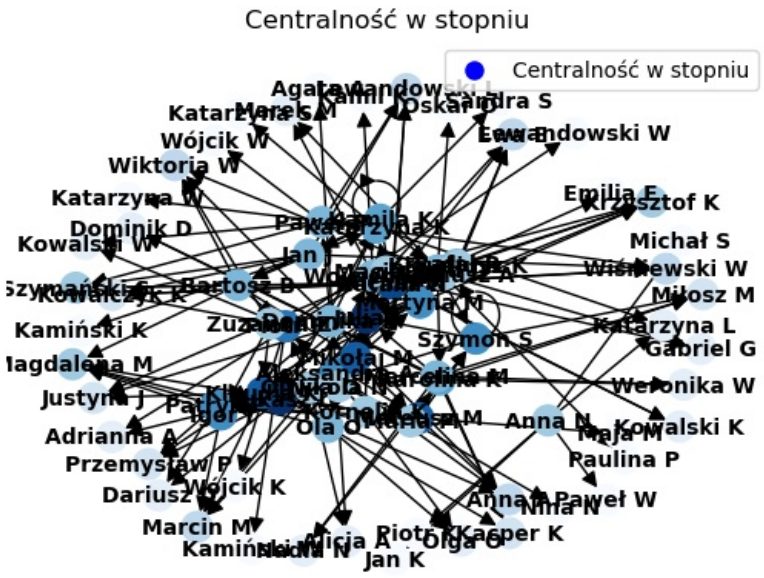
In [72]:

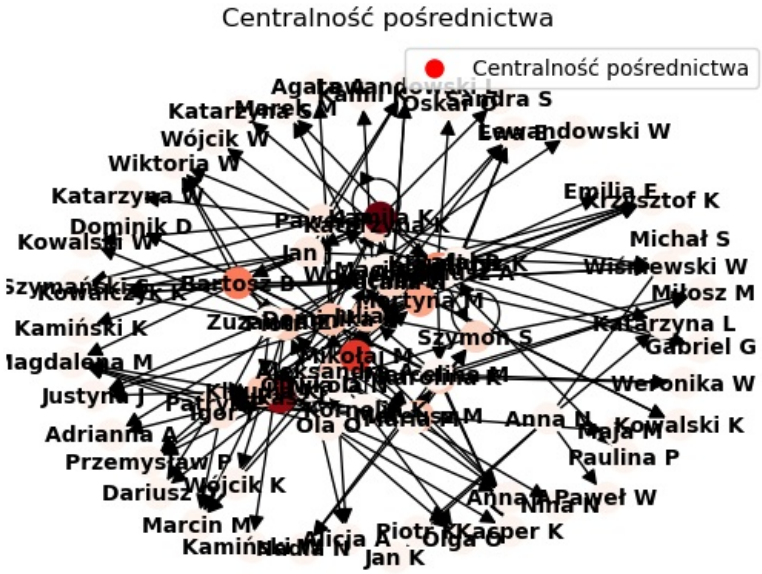
```python
import networkx as nx
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D

# Tworzenie skierowanego grafu
G = nx.DiGraph()

# Dodawanie krawędzi z wcześniejszych połączeń
edges = [

   ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
```

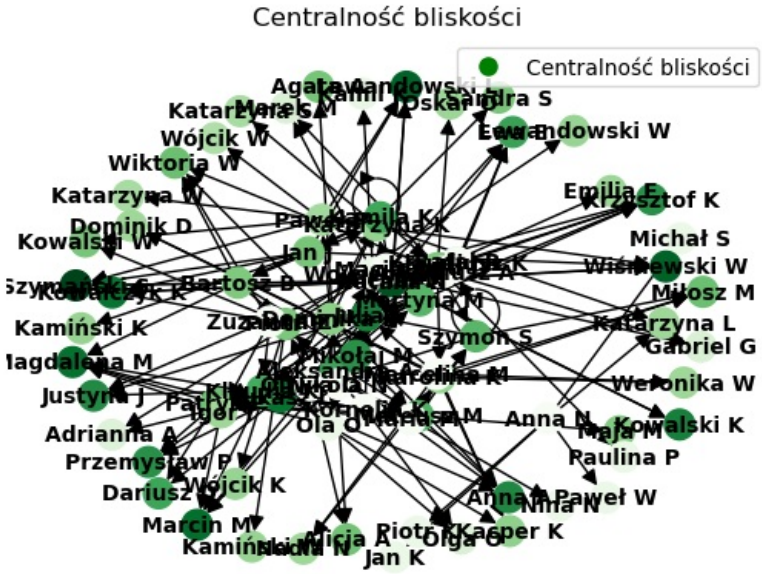
```
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
    ("Wojciech W", "Kamil K"),
    ("Wojciech W", "Adrianna A"),
    ("Wojciech W", "Marek M"),
    ("Wojciech W", "Gabriel G"),
    ("Wojciech W", "Julia J"),
    ("Karolina K", "Dominik D"),
    ("Karolina K", "Weronika W"),
    ("Karolina K", "Miłosz M"),
    ("Karolina K", "Nadia N"),
    ("Karolina K", "Kacper K"),
    ("Bartosz B", "Magdalena M"),
    ("Bartosz B", "Mikołaj M"),
    ("Bartosz B", "Wiktoria W"),
    ("Bartosz B", "Piotr P"),
    ("Bartosz B", "Katarzyna K"),
    ("Kamila K", "Jan J"),
    ("Kamila K", "Natalia N"),
    ("Kamila K", "Bartosz B"),
    ("Kamila K", "Sandra S"),
    ("Kamila K", "Dawid D"),
    ("Dawid D", "Emilia E"),
    ("Dawid D", "Szymon S"),
    ("Dawid D", "Karolina K"),
    ("Dawid D", "Oskar O"),
    ("Dawid D", "Klaudia K"),
    ("Dominika D", "Paweł P"),
    ("Dominika D", "Wiktoria W"),
    ("Dominika D", "Igor I"),
    ("Dominika D", "Anna A"),
    ("Dominika D", "Kamila K"),
    ("Patryk P", "Dariusz D"),
```

```
    ("Patryk P", "Magdalena M"),
    ("Patryk P", "Przemysław P"),
    ("Patryk P", "Aleksandra A"),
    ("Patryk P", "Mikołaj M"),
    ("Julia J", "Justyna J"),
    ("Julia J", "Marcin M"),
    ("Julia J", "Natalia N"),
    ("Julia J", "Krzysztof K"),
    ("Julia J", "Ewa E"),
    ("Mikołaj M", "Łukasz Ł"),
    ("Mikołaj M", "Martyna M"),
    ("Mikołaj M", "Michał M"),
    ("Mikołaj M", "Klaudia K"),
    ("Mikołaj M", "Mateusz M"),
    ("Ola O", "Julia J"),
    ("Ola O", "Piotr P"),
    ("Ola O", "Olga O"),
    ("Ola O", "Kacper K"),
    ("Ola O", "Magdalena M"),
    ("Maciej M", "Mikołaj M"),
    ("Maciej M", "Justyna J"),
    ("Maciej M", "Katarzyna K"),
    ("Maciej M", "Krzysztof K"),
    ("Maciej M", "Ewa E"),
    ("Nikola N", "Łukasz Ł"),
    ("Nikola N", "Klaudia K"),
    ("Nikola N", "Mateusz M"),
    ("Nikola N", "Julia J"),
    ("Nikola N", "Ola O"),
    ("Zuzanna Z", "Klaudia K"),
    ("Zuzanna Z", "Mateusz M"),
    ("Zuzanna Z", "Piotr P"),
    ("Zuzanna Z", "Wiktoria W"),
    ("Zuzanna Z", "Igor I"),
    ("Łukasz Ł", "Anna A"),
    ("Łukasz Ł", "Kamila K"),
    ("Łukasz Ł", "Dariusz D"),
    ("Łukasz Ł", "Magdalena M"),
    ("Łukasz Ł", "Przemysław P"),
    ("Klaudia K", "Aleksandra A"),
    ("Klaudia K", "Mikołaj M"),
    ("Klaudia K", "Justyna J"),
    ("Klaudia K", "Marcin M"),
    ("Klaudia K", "Natalia N"),
    ("Rafał R", "Krzysztof K"),
    ("Rafał R", "Ewa E"),
    ("Rafał R", "Łukasz Ł"),
    ("Rafał R", "Martyna M"),
    ("Rafał R", "Michał M"),
    ("Kornelia K", "Klaudia K"),
    ("Kornelia K", "Mateusz M"),
    ("Kornelia K", "Julia J"),
    ("Kornelia K", "Piotr P"),
    ("Kornelia K", "Olga O"),
    ("Arkadiusz A", "Natalia N"),
    ("Arkadiusz A", "Paulina P"),
    ("Arkadiusz A", "Wiktoria W"),
    ("Arkadiusz A", "Marek M"),
    ("Arkadiusz A", "Gabriel G"),
    ("Marcelina M", "Julia J"),
    ("Marcelina M", "Dominik D"),
    ("Marcelina M", "Weronika W"),
    ("Marcelina M", "Miłosz M"),
    ("Marcelina M", "Nadia N"),
    ("Marcelina M", "Kacper K"),
    ("Igor I", "Magdalena M"),
    ("Igor I", "Przemysław P"),
    ("Igor I", "Aleksandra A"),
    ("Igor I", "Mikołaj M"),
    ("Igor I", "Marcin M"),
    ("Klaudiusz K", "Natalia N"),
    ("Klaudiusz K", "Krzysztof K"),
    ("Klaudiusz K", "Ewa E"),
    ("Klaudiusz K", "Łukasz Ł"),
    ("Klaudiusz K", "Martyna M"),
    ("Martyna M", "Dominika D"),
    ("Martyna M", "Patryk P"),
    ("Martyna M", "Julia J"),
    ("Martyna M", "Natalia N"),
    ("Martyna M", "Krzysztof K"),
]

G.add_edges_from(edges)

# Obliczanie centralności w stopniu
degree_centrality = nx.degree_centrality(G)

# Obliczanie centralności pośrednictwa
betweenness_centrality = nx.betweenness_centrality(G)
```

```
# Obliczanie centralności bliskości
closeness_centrality = nx.closeness_centrality(G)

# Rysowanie grafu z oznaczeniem centralności kolorami
pos = nx.spring_layout(G)

# Centralność w stopniu
node_colors_degree = [degree_centrality[node] for node in G.nodes]
fig, ax = plt.subplots()
nx.draw(G, pos, with_labels=True, font_weight='bold', node_size=200, node_color=node_colors_degree, cmap=plt.cm
ax.set_title("Centralność w stopniu")

# Dodaj legendę
legend_elements = [Line2D([0], [0], marker='o', color='w', markerfacecolor='blue', markersize=10, label='Centra
ax.legend(handles=legend_elements, loc='upper right')

plt.show()

# Centralność pośrednictwa
node_colors_betweenness = [betweenness_centrality[node] for node in G.nodes]
fig, ax = plt.subplots()
nx.draw(G, pos, with_labels=True, font_weight='bold', node_size=200, node_color=node_colors_betweenness, cmap=p
ax.set_title("Centralność pośrednictwa")

# Dodaj legendę
legend_elements = [Line2D([0], [0], marker='o', color='w', markerfacecolor='red', markersize=10, label='Central
ax.legend(handles=legend_elements, loc='upper right')

plt.show()

# Centralność bliskości
node_colors_closeness = [closeness_centrality[node] for node in G.nodes]
fig, ax = plt.subplots()
nx.draw(G, pos, with_labels=True, font_weight='bold', node_size=200, node_color=node_colors_closeness, cmap=plt
ax.set_title("Centralność bliskości")

# Dodaj legendę
legend_elements = [Line2D([0], [0], marker='o', color='w', markerfacecolor='green', markersize=10, label='Centr
ax.legend(handles=legend_elements, loc='upper right')

plt.show()
```





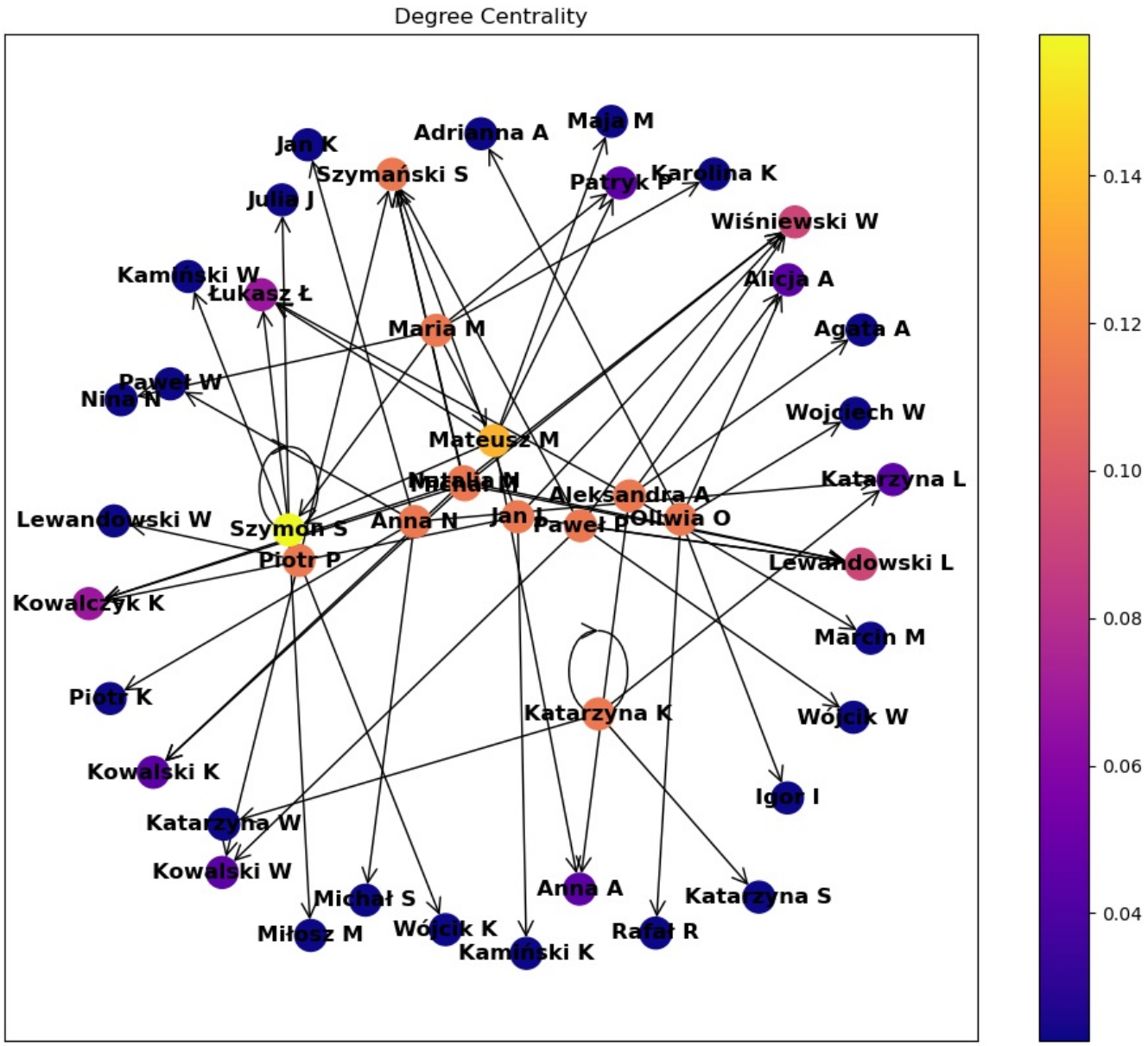


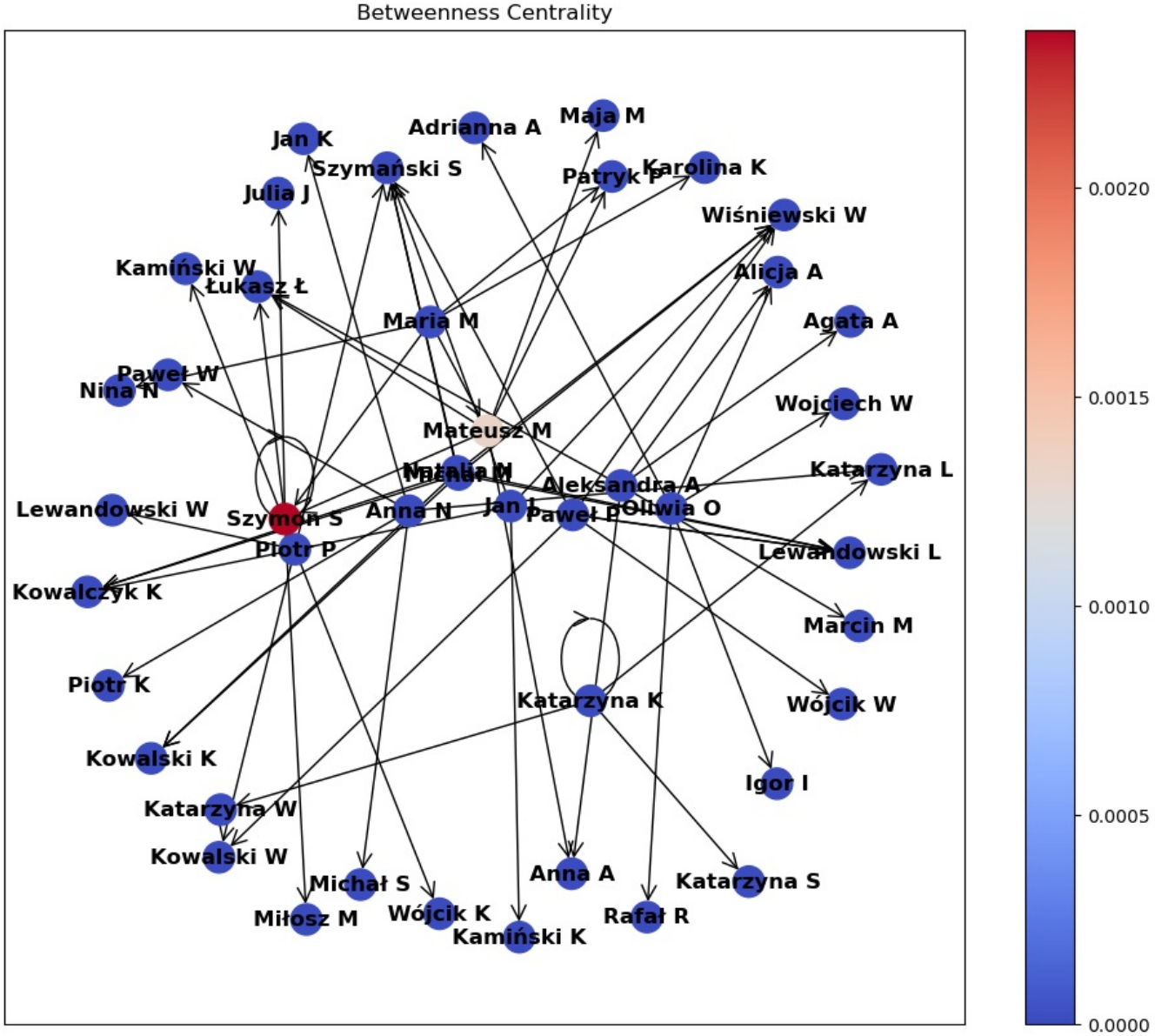
In [12]:

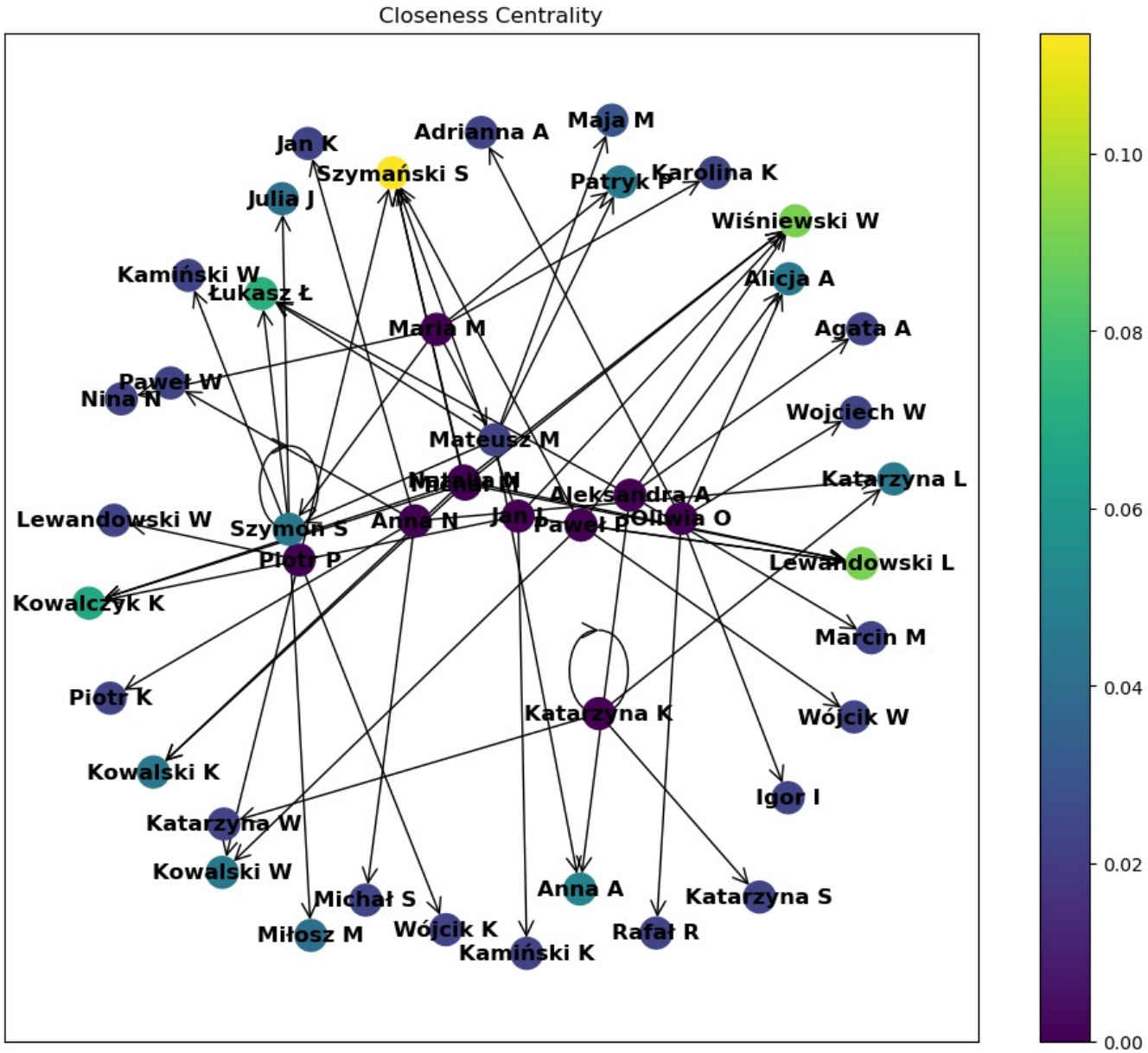
```python
import networkx as nx
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D

# Tworzenie skierowanego grafu
G = nx.DiGraph()

# Dodawanie krawędzi z wcześniejszych połączeń
edges = [
 ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
```

```
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
]

G.add_edges_from(edges)

# Obliczanie centralności w stopniu
degree_centrality = nx.degree_centrality(G)

# Obliczanie centralności pośrednictwa
betweenness_centrality = nx.betweenness_centrality(G)

# Obliczanie centralności bliskości
closeness_centrality = nx.closeness_centrality(G)

# Rysowanie grafu z oznaczeniem centralności kolorami
pos = nx.spring_layout(G)

# Centralność w stopniu
node_colors_degree = [degree_centrality[node] for node in G.nodes]
fig, ax = plt.subplots(figsize=(12, 10))
nodes = nx.draw_networkx_nodes(G, pos, node_color=node_colors_degree, cmap=plt.cm.plasma, node_size=300, ax=ax)
edges = nx.draw_networkx_edges(G, pos, arrowstyle='->', arrowsize=20, ax=ax)
nx.draw_networkx_labels(G, pos, font_color='black', font_size=12, font_weight='bold', ax=ax)
fig.colorbar(nodes, ax=ax)
ax.set_title("Degree Centrality")
plt.show()

# Centralność pośrednictwa
node_colors_betweenness = [betweenness_centrality[node] for node in G.nodes]
fig, ax = plt.subplots(figsize=(12, 10))
nodes = nx.draw_networkx_nodes(G, pos, node_color=node_colors_betweenness, cmap=plt.cm.coolwarm, node_size=300,
edges = nx.draw_networkx_edges(G, pos, arrowstyle='->', arrowsize=20, ax=ax)
nx.draw_networkx_labels(G, pos, font_color='black', font_size=12, font_weight='bold', ax=ax)
fig.colorbar(nodes, ax=ax)
ax.set_title("Betweenness Centrality")
plt.show()
```

```
# Centralność bliskości
node_colors_closeness = [closeness_centrality[node] for node in G.nodes]
fig, ax = plt.subplots(figsize=(12, 10))
nodes = nx.draw_networkx_nodes(G, pos, node_color=node_colors_closeness, cmap=plt.cm.viridis, node_size=300, ax
edges = nx.draw_networkx_edges(G, pos, arrowstyle='->', arrowsize=20, ax=ax)
nx.draw_networkx_labels(G, pos, font_color='black', font_size=12, font_weight='bold', ax=ax)
fig.colorbar(nodes, ax=ax)
ax.set_title("Closeness Centrality")
plt.show()
```



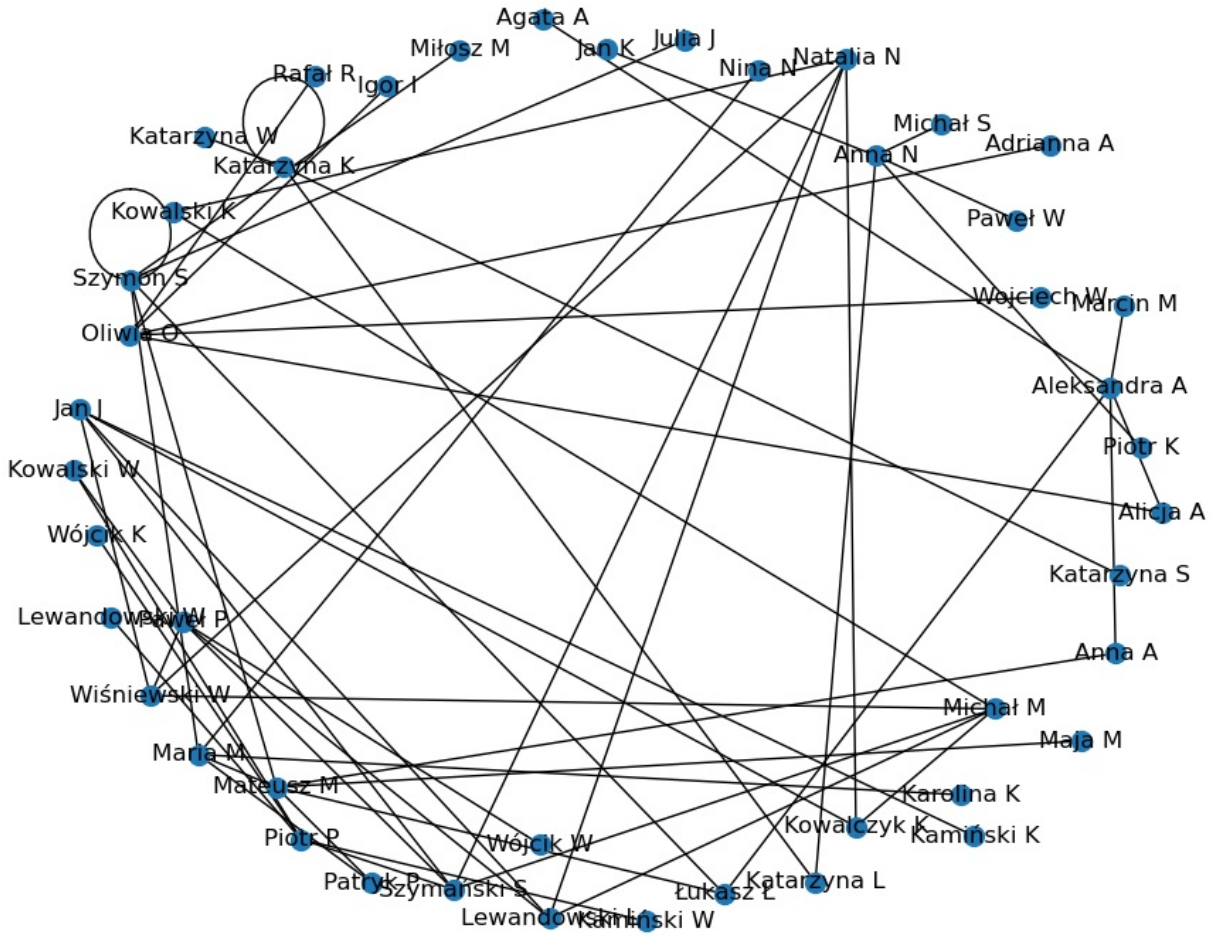

Betweenness Centrality
Jan K
Adrianna A
Maja M
Szymański S
Karolina K
Patryk P
Julia J
Wiśniewski W
Kamiński W
Łukasz Ł
Alicja A
Maria M
Agata A
Paweł W
Nina N
Wojciech W
Mateusz M
Katarzyna L
Aleksandra A
Lewandowski W
Szymon S
Anna N
Jan J
Paweł P
Oliwia O
Piotr P
Lewandowski L
Kowalczyk K
Marcin M
Piotr K
Katarzyna K
Wójcik W
Kowalski K
Igor I
Katarzyna W
Kowalski W
Anna A
Michał S
Katarzyna S
Miłosz M
Wójcik K
Rafał R
Kamiński K
0.0020
0.0015
0.0010
0.0005
0.0000



In [147…

```python
import networkx as nx
import matplotlib.pyplot as plt
from collections import Counter
import math

# Dane z wcześniejszych połączeń
connections = [
    ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
```

```
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
]

# Tworzenie grafu
G = nx.Graph()
G.add_edges_from(connections)

# Rysowanie grafu
pos = nx.spring_layout(G, k=3.0)
fig, ax = plt.subplots(figsize=(12, 10))
nx.draw(G, pos, with_labels=True,node_size=100)
plt.show()

# Funkcja do obliczania entropii
def calculate_entropy(connections):
    total_connections = len(connections)
    user_pairs = [f"{user1} - {user2}" for user1, user2 in connections]
    counter = Counter(user_pairs)
    probabilities = [count / total_connections for count in counter.values()]
    entropy = -sum(p * math.log2(p) for p in probabilities)
    return entropy

# Oblicz entropię dla danych
entropy_value = calculate_entropy(connections)
print(f'Entropy: {entropy_value:.2f}')
```



Entropy: 5.84

In [79]:

```
import networkx as nx

# Dane wcześniejszych połączeń
connections = [
      ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
```

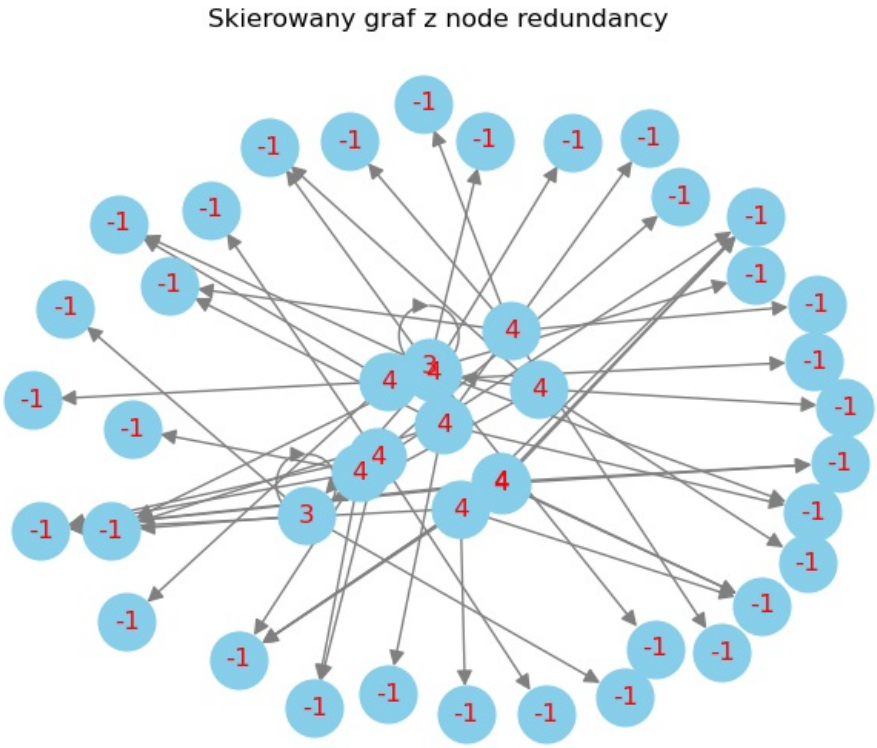
```
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
]

# Tworzenie grafu skierowanego
G = nx.DiGraph()
G.add_edges_from(connections)

# Obliczenie node redundancy
original_nodes = len(G.nodes)
minimum_nodes = nx.node_connectivity(G)

node_redundancy = original_nodes - minimum_nodes

print(f"Node Redundancy: {node_redundancy}")
```

```
Node Redundancy: 45
```

In [81]:

```
import networkx as nx
import matplotlib.pyplot as plt

# Dane wcześniejszych połączeń
connections = [
  ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
```

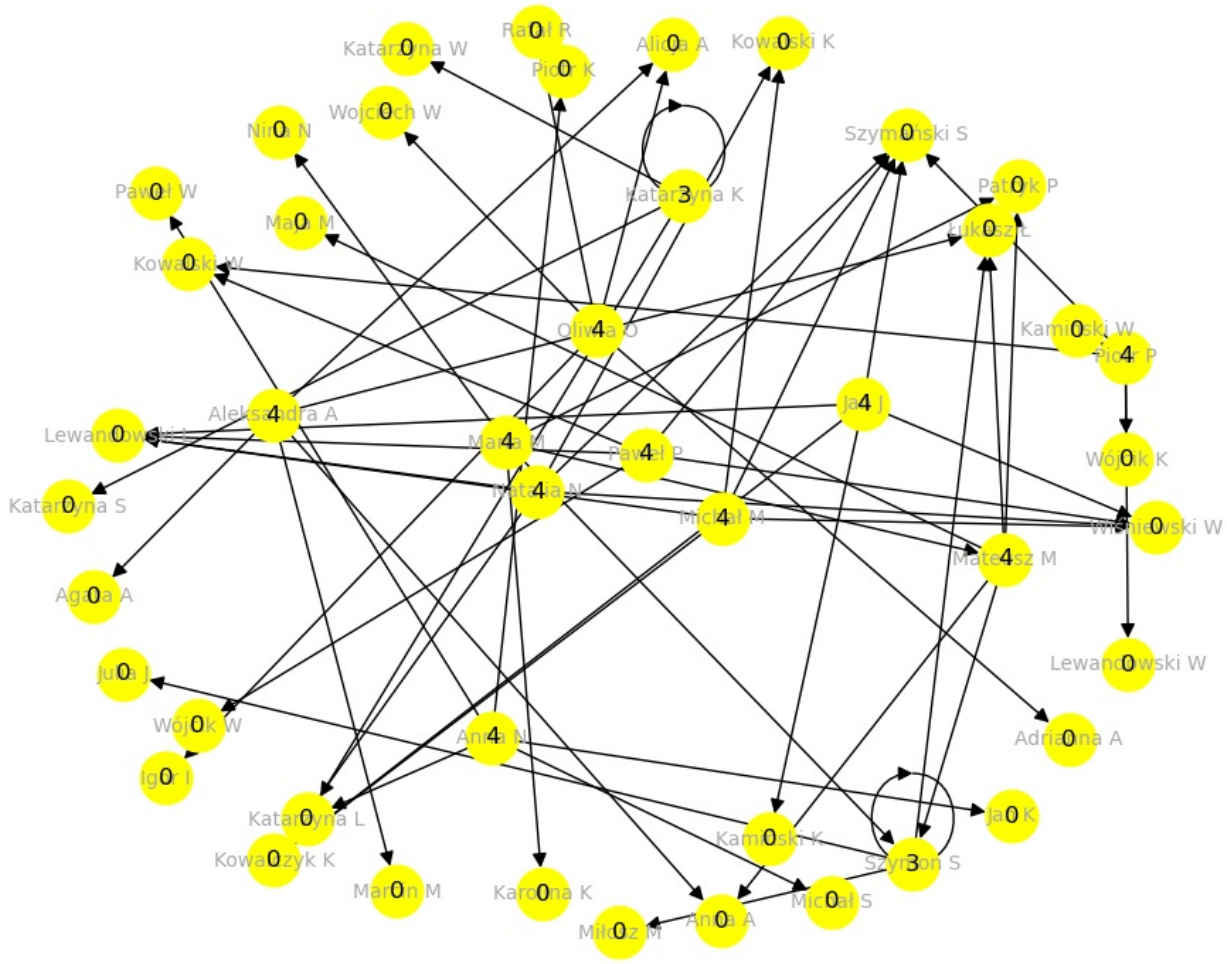
```
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
]

# Tworzenie grafu skierowanego
G = nx.DiGraph()
G.add_edges_from(connections)

# Rysowanie grafu
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=False, font_weight='bold', node_size=700, node_color='skyblue', edge_color='gray',

# Wizualizacja node redundancy jako etykiet na węzłach
node_redundancy_labels = {node: G.out_degree(node) - 1 for node in G.nodes}
nx.draw_networkx_labels(G, pos, labels=node_redundancy_labels, font_color='red')

plt.title('Skierowany graf z node redundancy')
plt.show()
```



In [173…

```
import networkx as nx
import matplotlib.pyplot as plt

# Dane wcześniejszych połączeń
connections = [
     ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
```

```
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
]

# Tworzenie grafu skierowanego
G = nx.DiGraph()
G.add_edges_from(connections)

# Rysowanie grafu
pos = nx.spring_layout(G, k=1.5)
fig = plt.subplots(figsize=(12, 10))
nx.draw(G, pos, with_labels=True, node_size=700, font_size=10, font_color='darkgray',node_color='yellow', edge_

# Wizualizacja node redundancy jako etykiet na węzłach
node_redundancy_labels = {node: max(0, G.out_degree(node) - 1) for node in G.nodes}
nx.draw_networkx_labels(G, pos, labels=node_redundancy_labels, font_color='black')

plt.title('Directed graph with node redundancy')
plt.show()
```



In [85]:
```
pip install python-louvain
```

```
Requirement already satisfied: python-louvain in c:\users\wlabl\anaconda3\lib\site-packages (0.16)
Requirement already satisfied: networkx in c:\users\wlabl\anaconda3\lib\site-packages (from python-louvain) (2.
8.8)
Requirement already satisfied: numpy in c:\users\wlabl\anaconda3\lib\site-packages (from python-louvain) (1.23.
5)
Note: you may need to restart the kernel to use updated packages.
```

In [86]:
```
import networkx as nx
import matplotlib.pyplot as plt
from community import community_louvain

# Dane wcześniejszych połączeń
connections = [
  ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
```

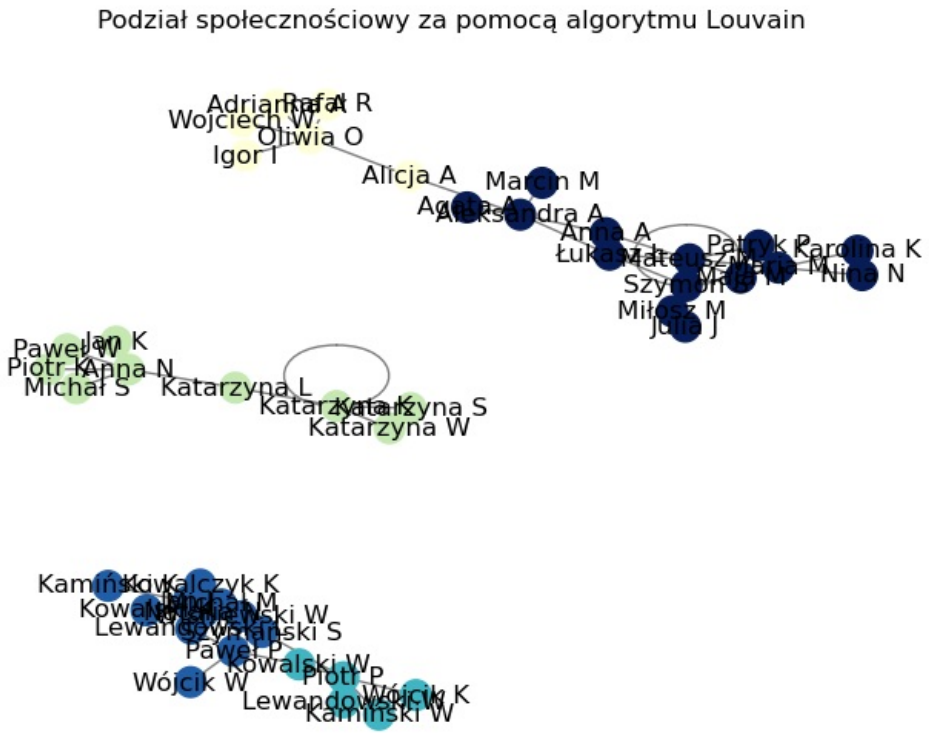
```
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
]

# Tworzenie grafu skierowanego
G = nx.DiGraph()
G.add_edges_from(connections)

# Algorytm Louvain
partition = community_louvain.best_partition(G)

# Rysowanie grafu z podziałem na społeczności
pos = nx.spring_layout(G)
colors = [partition[node] for node in G.nodes]
nx.draw(G, pos, with_labels=True, font_weight='bold', node_size=700, node_color=colors, cmap=plt.cm.get_cmap('v

plt.title('Podział na społeczności z algorytmem Louvain')
plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_9024\346788629.py in <module>
     72
     73 # Algorytm Louvain
---> 74 partition = community_louvain.best_partition(G)
     75
     76 # Rysowanie grafu z podziałem na społeczności

~\anaconda3\lib\site-packages\community\community_louvain.py in best_partition(graph, partition, weight, resolu
tion, randomize, random_state)
    247     >>> plt.show()
    248     """
--> 249     dendo = generate_dendrogram(graph,
    250                                 partition,
    251                                 weight,

~\anaconda3\lib\site-packages\community\community_louvain.py in generate_dendrogram(graph, part_init, weight, r
esolution, randomize, random_state)
    320     """
    321     if graph.is_directed():
--> 322         raise TypeError("Bad graph type, use only non directed graph")
    323
    324     # Properly handle random state, eventually remove old `randomize` parameter

TypeError: Bad graph type, use only non directed graph
```

In [1]:

```
import networkx as nx
import matplotlib.pyplot as plt
from community import community_louvain

# Dane wcześniejszych połączeń
connections = [
    ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
```

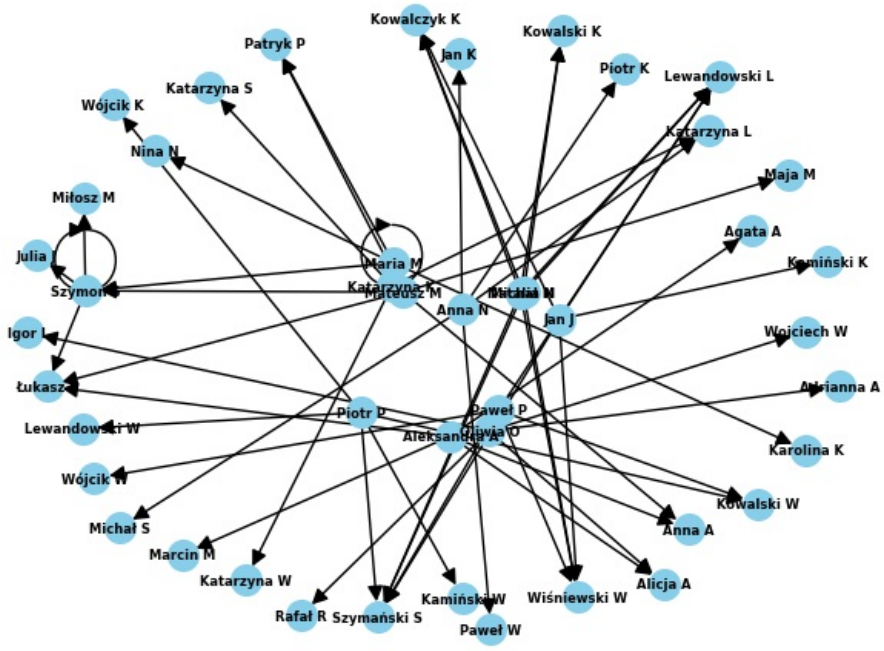
```
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
]

# Tworzenie grafu nieskierowanego
G = nx.Graph()
G.add_edges_from(connections)

# Algorytm Louvain
partition = community_louvain.best_partition(G)

# Rysowanie grafu z podziałem na społeczności
pos = nx.spring_layout(G)
colors = [partition[node] for node in G.nodes]
nx.draw(G, pos, with_labels=True,  node_size=200, node_color=colors, cmap=plt.cm.get_cmap('YlGnBu'), edge_color

plt.title('Podział społecznościowy za pomocą algorytmu Louvain ')
plt.show()
```



In [16]:

```python
import networkx as nx
import matplotlib.pyplot as plt

# Tworzenie skierowanego grafu
G = nx.DiGraph()

# Dodawanie krawędzi z wcześniejszych połączeń
edges = [
    ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
```

```
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
]

G.add_edges_from(edges)

# Obliczanie współczynnika klastrowania dla całego grafu
clustering_coefficient = nx.average_clustering(G)

# Rysowanie grafu z większymi wierzchołkami
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, font_weight='bold', node_size=200, node_color='skyblue', font_color='black',

# Wyświetlanie grafu
plt.show()

# Wyświetlanie współczynnika klastrowania dla całego grafu
print(f'Średni współczynnik klastrowania dla całego grafu: {clustering_coefficient:.2f}')
```

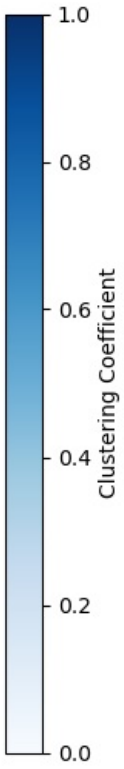



```
Średni współczynnik klastrowania dla całego grafu: 0.02
```

In [100…

```
import networkx as nx
import matplotlib.pyplot as plt

# Tworzenie skierowanego grafu
G = nx.DiGraph()

# Dodawanie krawędzi z wcześniejszych połączeń
edges = [
    # ... (jak wcześniej)
]

G.add_edges_from(edges)

# Obliczanie współczynnika klastrowania dla każdego wierzchołka
clustering_coefficients = nx.clustering(G)

# Rysowanie grafu z różnymi kolorami w zależności od współczynnika klastrowania
pos = nx.spring_layout(G)
node_colors = [clustering_coefficients[node] for node in G.nodes]
nx.draw(G, pos, with_labels=True, font_weight='bold', node_size=200, node_color=node_colors, cmap=plt.cm.Blues,

# Dodanie kolorowej skali do grafu
sm = plt.cm.ScalarMappable(cmap=plt.cm.Blues)
sm.set_array([])
cbar = plt.colorbar(sm, orientation='vertical')
```

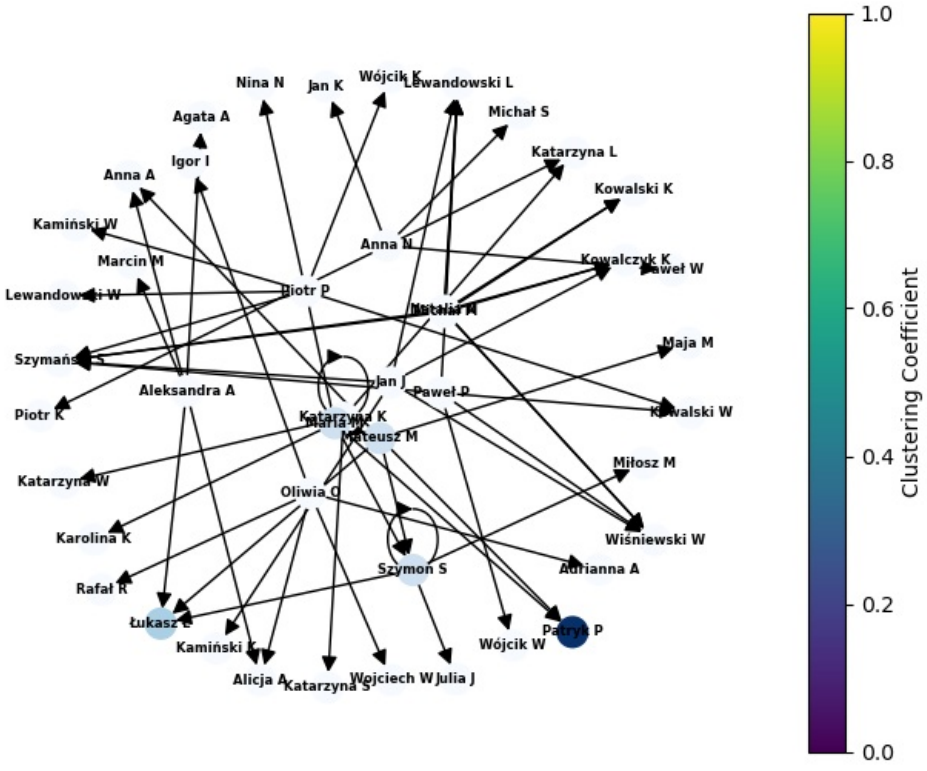
```
cbar.set_label('Clustering Coefficient')

# Wyświetlanie grafu
plt.show()
```



In [17]:

```
import networkx as nx
import matplotlib.pyplot as plt

# Tworzenie skierowanego grafu
G = nx.DiGraph()

# Dodawanie krawędzi z wcześniejszych połączeń
edges = [
    ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
```

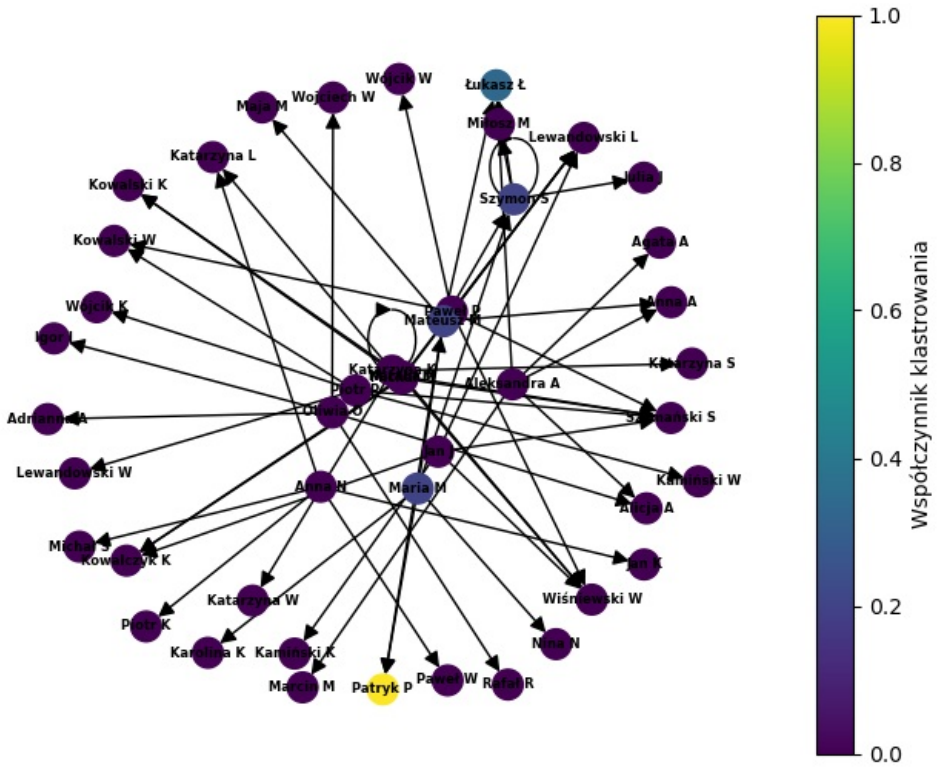
```
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
]

G.add_edges_from(edges)

# Obliczanie współczynnika klastrowania dla każdego wierzchołka
clustering_coefficients = nx.clustering(G)

# Rysowanie grafu z różnymi kolorami w zależności od współczynnika klastrowania
pos = nx.spring_layout(G)
node_colors = [clustering_coefficients[node] for node in G.nodes]
nx.draw(G, pos, with_labels=True, font_weight='bold', node_size=200, node_color=node_colors, cmap=plt.cm.Blues,

# Dodanie kolorowej skali do grafu
sm = plt.cm.ScalarMappable(cmap=plt.cm.viridis)
sm.set_array([])
cbar = plt.colorbar(sm, orientation='vertical')
cbar.set_label('Clustering Coefficient')

# Wyświetlanie grafu
plt.show()
```



In [4]:

```
import networkx as nx
import matplotlib.pyplot as plt

# Tworzenie skierowanego grafu
G = nx.DiGraph()

# Dodawanie krawędzi z wcześniejszych połączeń
edges = [
 ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
```

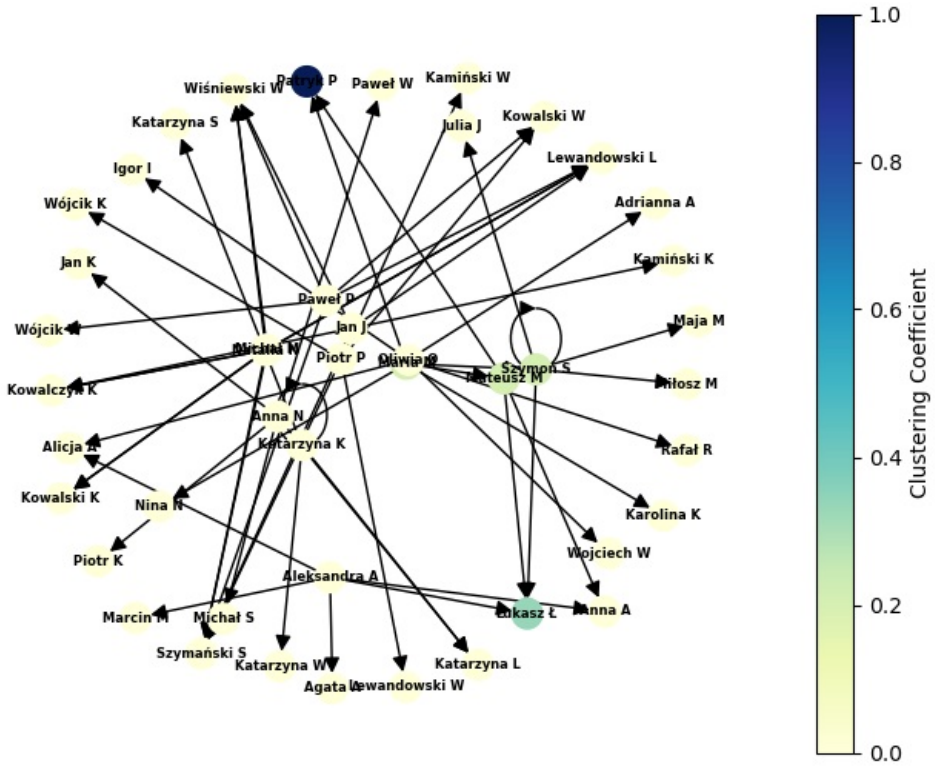
```
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
]

G.add_edges_from(edges)

# Obliczanie współczynnika klastrowania dla każdego wierzchołka
clustering_coefficients = nx.clustering(G)

# Rysowanie grafu z różnymi kolorami w zależności od współczynnika klastrowania
pos = nx.spring_layout(G)
node_colors = [clustering_coefficients[node] for node in G.nodes]
nx.draw(G, pos, with_labels=True, font_weight='bold', node_size=200, node_color=node_colors, cmap=plt.cm.viridi

# Dodanie kolorowej skali do grafu
sm = plt.cm.ScalarMappable(cmap=plt.cm.viridis)
sm.set_array([])
cbar = plt.colorbar(sm, orientation='vertical')
cbar.set_label('Współczynnik klastrowania')

# Wyświetlanie grafu
plt.show()
```



In [107…

```python
import networkx as nx
import matplotlib.pyplot as plt

# Tworzenie skierowanego grafu
G = nx.DiGraph()

# Dodawanie krawędzi z wcześniejszych połączeń
edges = [
("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
```

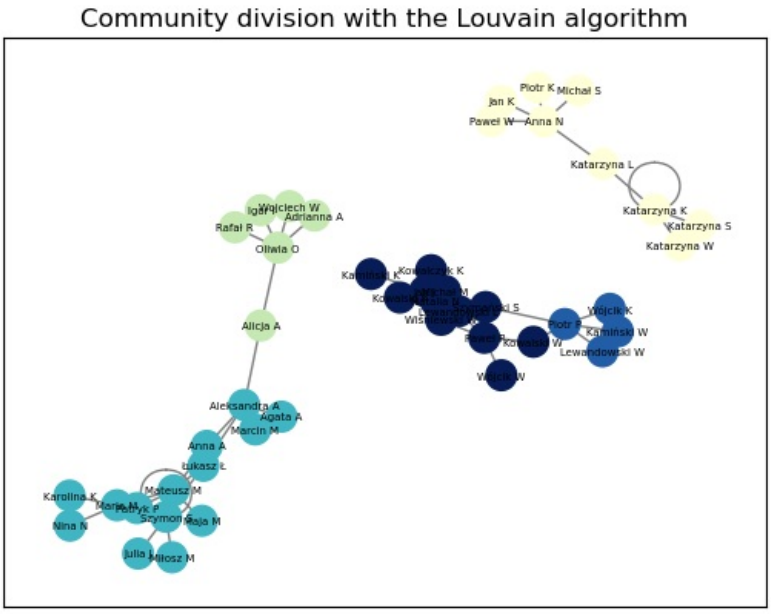
```
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
]

G.add_edges_from(edges)

# Obliczanie współczynnika klastrowania dla każdego wierzchołka
clustering_coefficients = nx.clustering(G)

# Rysowanie grafu z różnymi kolorami w zależności od współczynnika klastrowania
pos = nx.spring_layout(G)
node_colors = [clustering_coefficients[node] for node in G.nodes]
nx.draw(G, pos, with_labels=True, font_weight='bold', node_size=200, node_color=node_colors, cmap=plt.cm.YlGnBu

# Dodanie kolorowej skali do grafu
sm = plt.cm.ScalarMappable(cmap=plt.cm.YlGnBu)
sm.set_array([])
cbar = plt.colorbar(sm, orientation='vertical')
cbar.set_label('Clustering Coefficient')

# Wyświetlanie grafu
plt.show()
```



In [4]:

```
import networkx as nx
import matplotlib.pyplot as plt
from community import community_louvain

# Dane wcześniejszych połączeń
connections = [
  ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
```

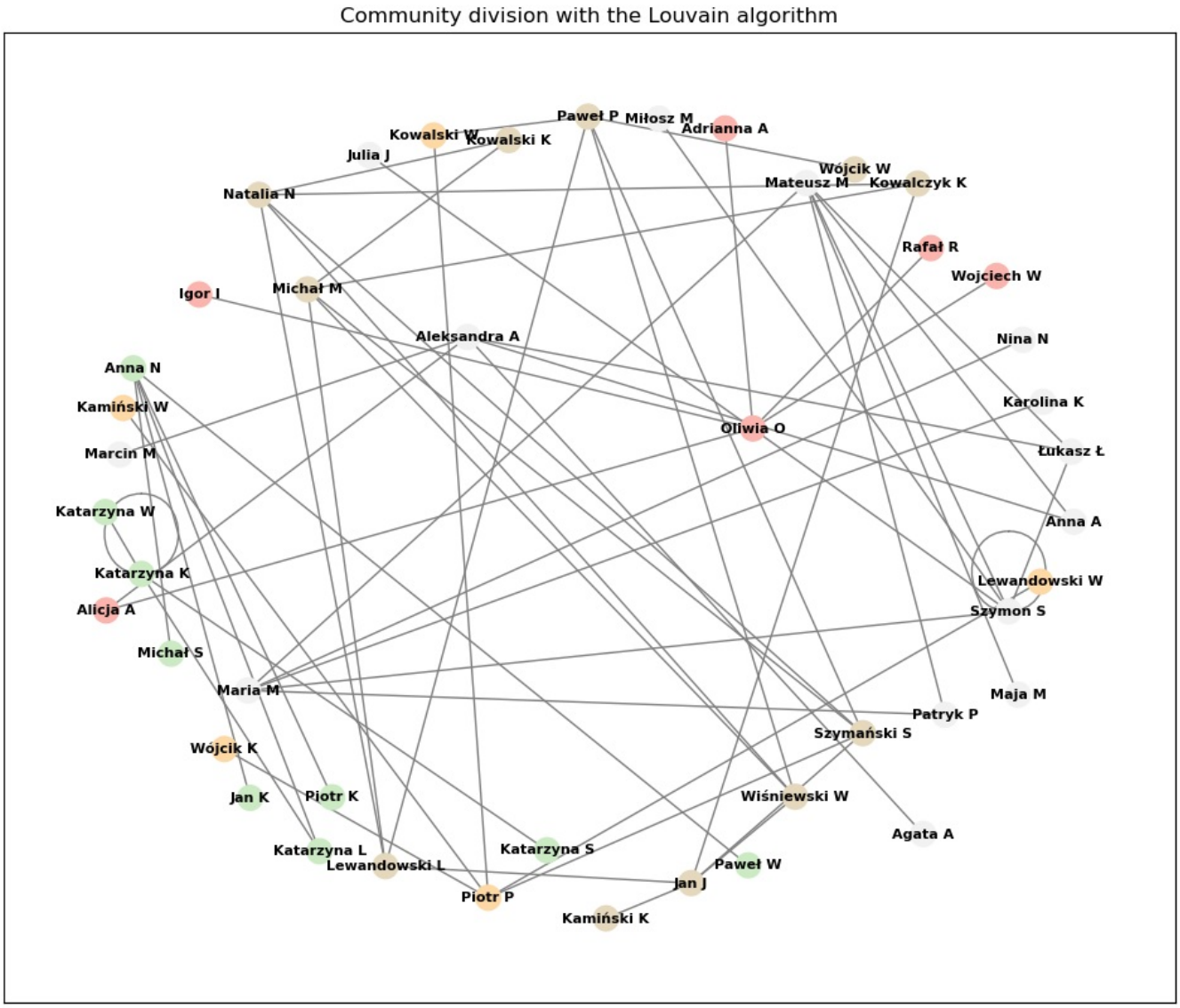
```
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
]

# Tworzenie grafu nieskierowanego
G = nx.Graph()
G.add_edges_from(connections)

# Algorytm Louvain
partition = community_louvain.best_partition(G)

# Rysowanie grafu z podziałem na społeczności
pos = nx.spring_layout(G)
colors = [partition[node] for node in G.nodes]

# Rysowanie wierzchołków
nx.draw_networkx_nodes(G, pos, node_color=colors, cmap=plt.cm.get_cmap('YlGnBu'), node_size=200)

# Rysowanie krawędzi
nx.draw_networkx_edges(G, pos, edge_color='gray')

# Dodanie etykiet wierzchołków nad wierzchołkami
nx.draw_networkx_labels(G, pos, labels={node: node for node in G.nodes},  font_size=5, font_color='black')

plt.title('Community division with the Louvain algorithm')
plt.show()
```



In [130…

```
import networkx as nx
import matplotlib.pyplot as plt
from community import community_louvain

# Dane wcześniejszych połączeń
connections = [
     ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
```

```
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
]

# Tworzenie grafu nieskierowanego
G = nx.Graph()
G.add_edges_from(connections)

# Algorytm Louvain
partition = community_louvain.best_partition(G)

# Rysowanie grafu z podziałem na społeczności
pos = nx.spring_layout(G, k=2.1)  # Dostosuj parametr k według potrzeb
colors = [partition[node] for node in G.nodes]
fig, ax = plt.subplots(figsize=(12, 10))

# Rysowanie wierzchołków
nx.draw_networkx_nodes(G, pos, node_color=colors, cmap=plt.cm.get_cmap('Pastel1'), node_size=200)

# Rysowanie krawędzi
nx.draw_networkx_edges(G, pos, edge_color='gray')

# Dodanie etykiet wierzchołków nad wierzchołkami
nx.draw_networkx_labels(G, pos, labels={node: node for node in G.nodes}, font_size=8, font_color='black', font_
```



In [42]:

```
import networkx as nx

# Dane wcześniejszych połączeń
connections = [
    ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
```

```
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
]

# Tworzenie grafu nieskierowanego
G = nx.Graph()
G.add_edges_from(connections)

# Obliczanie średniego stopnia wierzchołka
N = len(G.nodes)
M = len(G.edges)
average_degree = (2 * M) / N

print(f"Average degree of a vertex: {average_degree}")
```

```
Average degree of a vertex: 2.577777777777778
```

In [43]:

```
import networkx as nx

# Dane wcześniejszych połączeń
connections = [
 ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
```

```
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
]

# Tworzenie grafu nieskierowanego
G = nx.Graph()
G.add_edges_from(connections)

# Obliczanie średnicy sieci
diameter = nx.diameter(G)

print(f"Network diameter: {diameter}")
```

```
---------------------------------------------------------------------------
NetworkXError                             Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_15996\3707947431.py in <module>
     70
     71 # Obliczanie średnicy sieci
---> 72 diameter = nx.diameter(G)
     73
     74 print(f"Network diameter: {diameter}")

~\anaconda3\lib\site-packages\networkx\algorithms\distance_measures.py in diameter(G, e, usebounds)
    386         return _extrema_bounding(G, compute="diameter")
    387     if e is None:
--> 388         e = eccentricity(G)
    389     return max(e.values())
    390

~\anaconda3\lib\site-packages\networkx\algorithms\distance_measures.py in eccentricity(G, v, sp)
    345             else:
    346                 msg = "Found infinite path length because the graph is not" " connected"
--> 347             raise nx.NetworkXError(msg)
    348
    349         e[n] = max(length.values())

NetworkXError: Found infinite path length because the graph is not connected
```

In [44]:

```
import networkx as nx

# Dane wcześniejszych połączeń
connections = [
   ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
```

```
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
]

# Tworzenie grafu nieskierowanego
G = nx.Graph()
G.add_edges_from(connections)

# Sprawdź czy graf jest spójny
if nx.is_connected(G):
    # Jeśli jest spójny, oblicz średnicę sieci
    diameter = nx.diameter(G)
    print(f"Network diameter: {diameter}")
else:
    print("Graph is not connected, diameter is infinite.")
```

```
Graph is not connected, diameter is infinite.
```

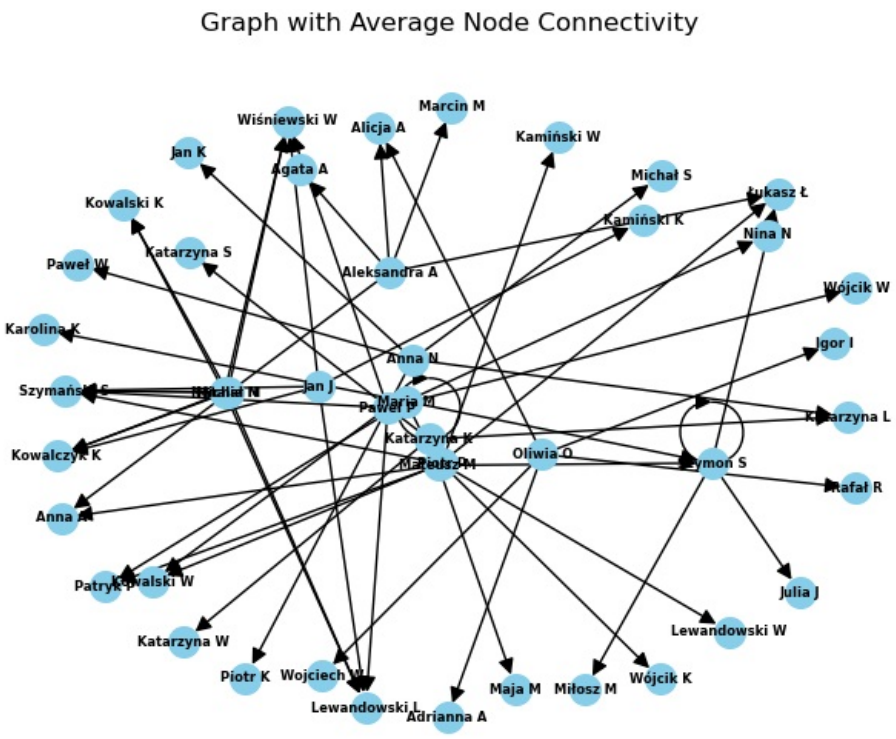
In [71]:
```
nx.average_node_connectivity(graph, flow_func=None)
```

Out[71]:
```
0.47342995169082125
```

In [70]:
```
import networkx as nx
import matplotlib.pyplot as plt

# Dane wcześniejszych połączeń
connections = [
    ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
```

```
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A")

]

# Tworzenie grafu skierowanego
G = nx.DiGraph()
G.add_edges_from(connections)

# Oblicz średnią zdolność łączności węzłów
average_connectivity = nx.average_node_connectivity(G)
print(f"Average node connectivity: {average_connectivity}")

# Rysuj graf
pos = nx.spring_layout(G, seed=42)  # Seed dla reprodukowalności układu wierzchołków
nx.draw(G, pos, with_labels=True, font_weight='bold', node_size=200, node_color='skyblue', font_color='black',

plt.title('Graph with Average Node Connectivity')
plt.show()
```

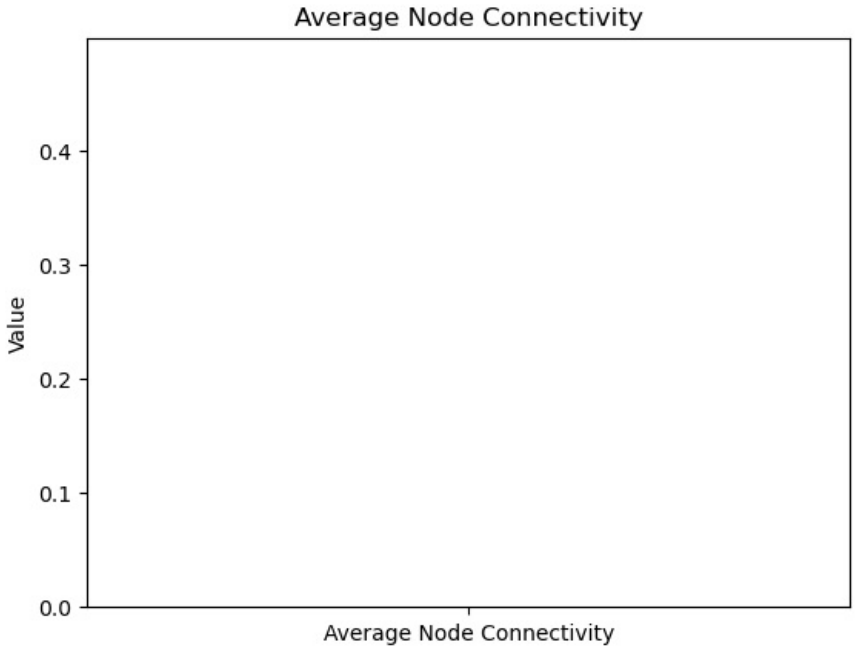
```
Average node connectivity: 0.03383838383838384
```



In [73]:

```
import networkx as nx
import matplotlib.pyplot as plt

# Dane wcześniejszych połączeń
connections = [
   ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
```

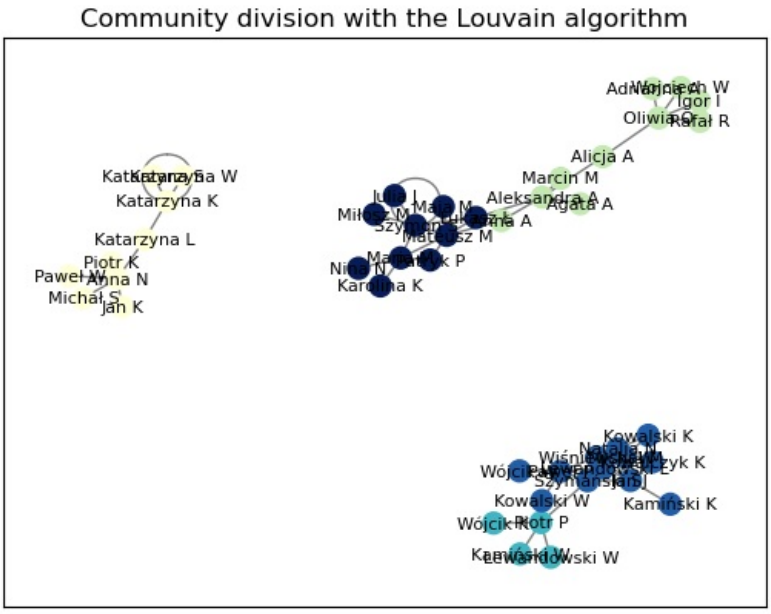
```
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A")

]

# Tworzenie grafu nieskierowanego
G = nx.Graph()
G.add_edges_from(connections)

# Obliczanie średniego stopnia wierzchołka
average_node_connectivity = nx.average_node_connectivity(G)

# Rysowanie wykresu
plt.bar(['Average Node Connectivity'], [average_node_connectivity], color='white')
plt.title('Average Node Connectivity')
plt.ylabel('Value')
plt.show()
```



In [6]:

```
import networkx as nx
import matplotlib.pyplot as plt
from community import community_louvain

# Dane wcześniejszych połączeń
connections = [
    ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
```

```
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
]

# Tworzenie grafu nieskierowanego
G = nx.Graph()
G.add_edges_from(connections)

# Algorytm Louvain
partition = community_louvain.best_partition(G)

# Rysowanie grafu z podziałem na społeczności
pos = nx.spring_layout(G)
colors = [partition[node] for node in G.nodes]

# Rysowanie wierzchołków z dostosowanym rozmiarem
node_size = 100  # Dostosuj według potrzeb
nx.draw_networkx_nodes(G, pos, node_color=colors, cmap=plt.cm.get_cmap('YlGnBu'), node_size=node_size)

# Rysowanie krawędzi
nx.draw_networkx_edges(G, pos, edge_color='gray')

# Dodanie etykiet wierzchołków nad wierzchołkami
nx.draw_networkx_labels(G, pos, labels={node: node for node in G.nodes}, font_size=8, font_color='black')

plt.title('Community division with the Louvain algorithm')
plt.show()
```



In [8]:

```python
import numpy as np

# Lista wierzchołków
vertices = ["Anna N", "Katarzyna L", "Piotr K", "Paweł W", "Michał S", "Jan K", "Katarzyna W", "Katarzyna S", "

# Lista krawędzi
edges = [
     ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
```

```
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
]

# Tworzenie macierzy sąsiedztwa
num_vertices = len(vertices)
adjacency_matrix = np.zeros((num_vertices, num_vertices), dtype=int)

for edge in edges:
    idx1 = vertices.index(edge[0])
    idx2 = vertices.index(edge[1])
    adjacency_matrix[idx1, idx2] = 1
    adjacency_matrix[idx2, idx1] = 1

# Wyświetlanie macierzy sąsiedztwa
print("Adjacency Matrix:")
print(adjacency_matrix)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_1248\724726569.py in <module>
     73
     74 for edge in edges:
---> 75     idx1 = vertices.index(edge[0])
     76     idx2 = vertices.index(edge[1])
     77     adjacency_matrix[idx1, idx2] = 1

ValueError: 'Paweł P' is not in list
```

In [9]:

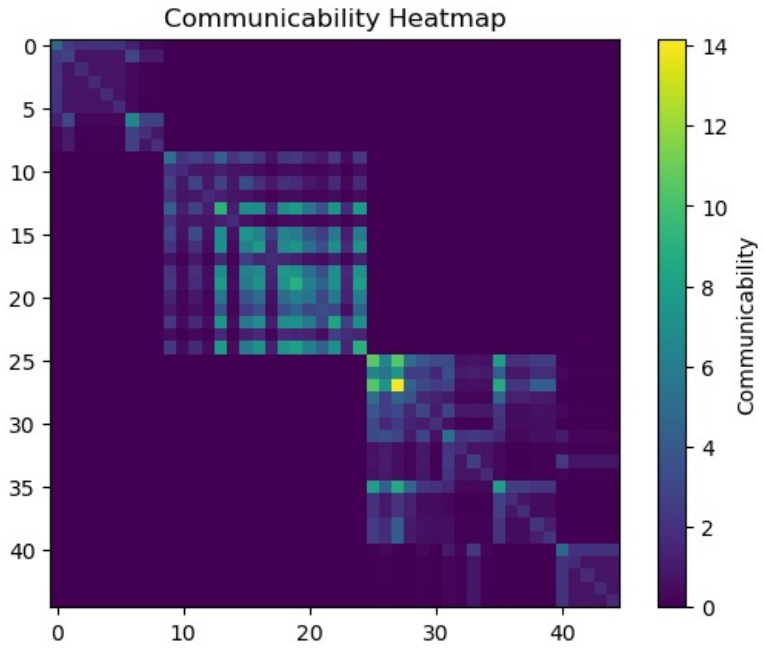
```
import numpy as np

# Lista wierzchołków
vertices = ["Anna N", "Katarzyna L", "Piotr K", "Paweł W", "Michał S", "Jan K", "Katarzyna W", "Katarzyna S", "

# Lista krawędzi
edges = [
       ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
```

```
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
]

# Tworzenie macierzy sąsiedztwa
num_vertices = len(vertices)
adjacency_matrix = np.zeros((num_vertices, num_vertices), dtype=int)

for edge in edges:
    if edge[0] in vertices and edge[1] in vertices:
        idx1 = vertices.index(edge[0])
        idx2 = vertices.index(edge[1])
        adjacency_matrix[idx1, idx2] = 1
        adjacency_matrix[idx2, idx1] = 1

# Wyświetlanie macierzy sąsiedztwa
print("Adjacency Matrix:")
print(adjacency_matrix)
```

```
Adjacency Matrix:
[[0 1 1 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

In [22]:

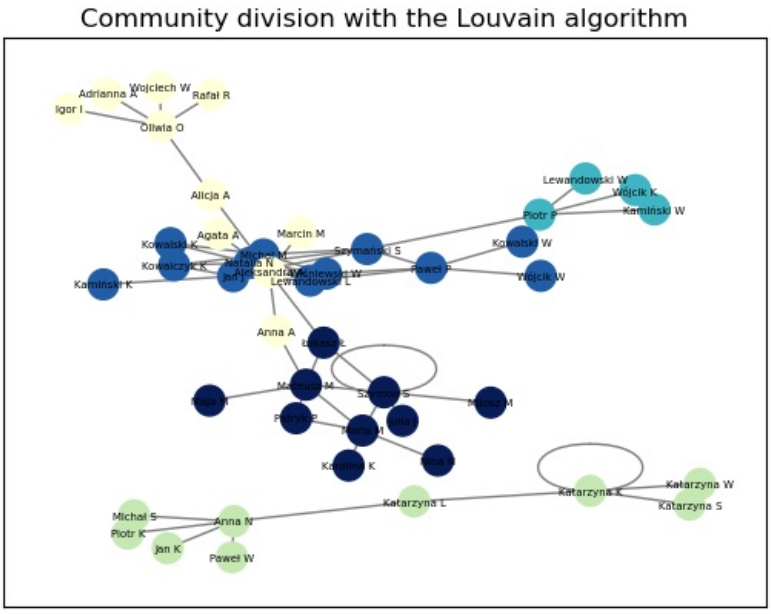
```
import numpy as np
import matplotlib.pyplot as plt

# Wizualizacja macierzy komunikowalności
communicability_matrix = nx.communicability(G)
communicability_array = np.array([[value for value in row.values()] for row in communicability_matrix.values()]

plt.imshow(communicability_array, cmap='viridis')
plt.colorbar(label="Communicability")
plt.title("Communicability Heatmap")
plt.show()
```

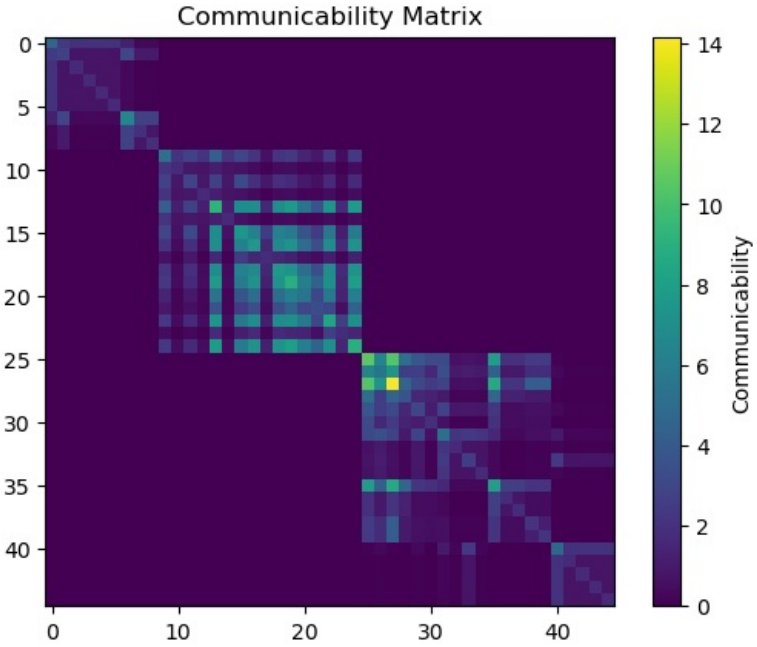


In [3]:

```
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
from community import community_louvain

# Dane wcześniejszych połączeń
connections = [
      ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
```

```
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
]

# Tworzenie grafu nieskierowanego
G = nx.Graph()
G.add_edges_from(connections)

# Algorytm Louvain
partition = community_louvain.best_partition(G)

# Rysowanie grafu z podziałem na społeczności
pos = nx.spring_layout(G)
colors = [partition[node] for node in G.nodes]

# Rysowanie wierzchołków
nx.draw_networkx_nodes(G, pos, node_color=colors, cmap=plt.cm.get_cmap('YlGnBu'), node_size=200)

# Rysowanie krawędzi
nx.draw_networkx_edges(G, pos, edge_color='gray')

# Dodanie etykiet wierzchołków nad wierzchołkami
nx.draw_networkx_labels(G, pos, labels={node: node for node in G.nodes}, font_size=5, font_color='black')

plt.title('Community division with the Louvain algorithm')
plt.show()

# Wizualizacja macierzy komunikowalności
communicability_matrix = nx.communicability(G)
communicability_array = np.array([[value for value in row.values()] for row in communicability_matrix.values()])

plt.imshow(communicability_array, cmap='viridis')
plt.colorbar(label="Communicability")
plt.title("Communicability Matrix")
plt.show()
```



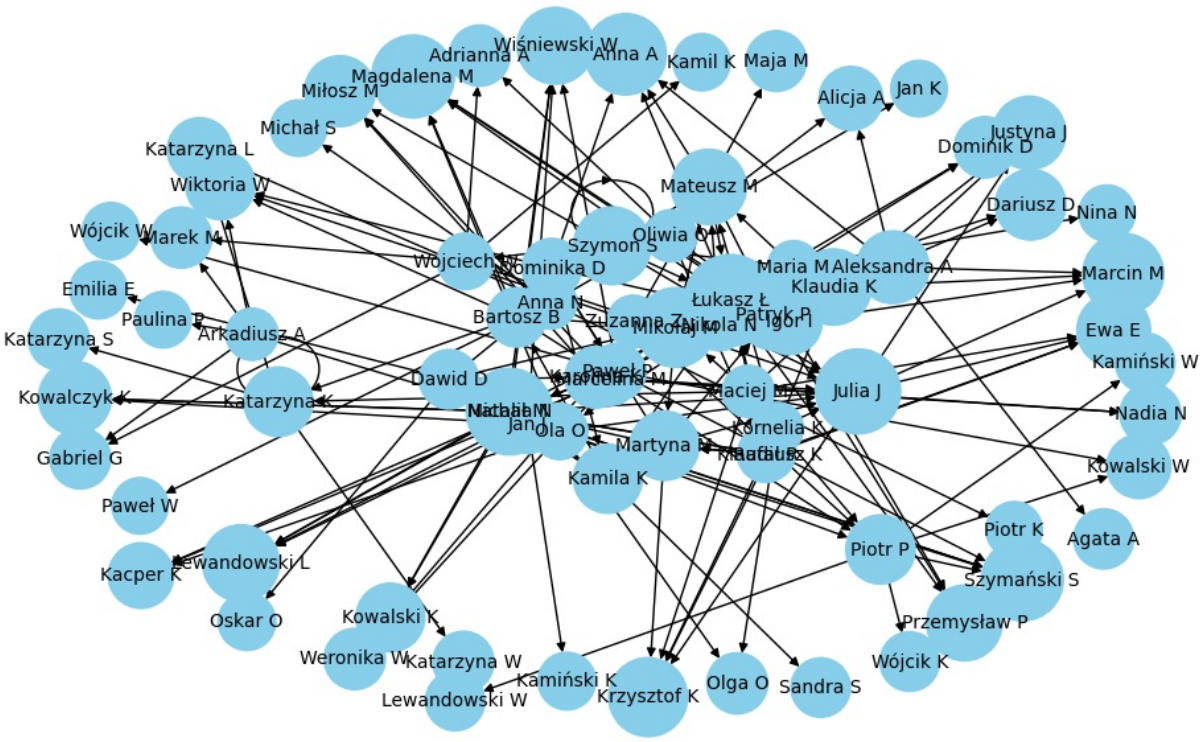


In [7]:

```python
import networkx as nx
import matplotlib.pyplot as plt

# Tworzenie skierowanego grafu
G = nx.DiGraph()

# Dodawanie krawędzi z wcześniejszych połączeń
edges = [

   ("Anna N", "Katarzyna L"),
    ("Anna N", "Piotr K"),
    ("Anna N", "Paweł W"),
    ("Anna N", "Michał S"),
    ("Anna N", "Jan K"),
    ("Katarzyna K", "Katarzyna W"),
    ("Katarzyna K", "Katarzyna L"),
    ("Katarzyna K", "Katarzyna K"),
    ("Katarzyna K", "Katarzyna S"),
    ("Katarzyna K", "Katarzyna K"),
    ("Piotr P", "Lewandowski W"),
    ("Piotr P", "Kowalski W"),
    ("Piotr P", "Kamiński W"),
    ("Piotr P", "Szymański S"),
    ("Piotr P", "Wójcik K"),
    ("Paweł P", "Wiśniewski W"),
    ("Paweł P", "Kowalski W"),
    ("Paweł P", "Wójcik W"),
    ("Paweł P", "Szymański S"),
    ("Paweł P", "Lewandowski L"),
    ("Michał M", "Wiśniewski W"),
    ("Michał M", "Lewandowski L"),
    ("Michał M", "Kowalczyk K"),
    ("Michał M", "Szymański S"),
    ("Michał M", "Kowalski K"),
```

```
    ("Jan J", "Lewandowski L"),
    ("Jan J", "Wiśniewski W"),
    ("Jan J", "Kamiński K"),
    ("Jan J", "Szymański S"),
    ("Jan J", "Kowalczyk K"),
    ("Natalia N", "Lewandowski L"),
    ("Natalia N", "Kowalski K"),
    ("Natalia N", "Wiśniewski W"),
    ("Natalia N", "Szymański S"),
    ("Natalia N", "Kowalczyk K"),
    ("Mateusz M", "Łukasz Ł"),
    ("Mateusz M", "Szymon S"),
    ("Mateusz M", "Patryk P"),
    ("Mateusz M", "Anna A"),
    ("Mateusz M", "Maja M"),
    ("Aleksandra A", "Łukasz Ł"),
    ("Aleksandra A", "Anna A"),
    ("Aleksandra A", "Agata A"),
    ("Aleksandra A", "Alicja A"),
    ("Aleksandra A", "Marcin M"),
    ("Maria M", "Karolina K"),
    ("Maria M", "Mateusz M"),
    ("Maria M", "Szymon S"),
    ("Maria M", "Nina N"),
    ("Maria M", "Patryk P"),
    ("Szymon S", "Julia J"),
    ("Szymon S", "Szymon S"),
    ("Szymon S", "Miłosz M"),
    ("Szymon S", "Łukasz Ł"),
    ("Szymon S", "Łukasz Ł"),
    ("Oliwia O", "Adrianna A"),
    ("Oliwia O", "Wojciech W"),
    ("Oliwia O", "Igor I"),
    ("Oliwia O", "Rafał R"),
    ("Oliwia O", "Alicja A"),
    ("Wojciech W", "Kamil K"),
    ("Wojciech W", "Adrianna A"),
    ("Wojciech W", "Marek M"),
    ("Wojciech W", "Gabriel G"),
    ("Wojciech W", "Julia J"),
    ("Karolina K", "Dominik D"),
    ("Karolina K", "Weronika W"),
    ("Karolina K", "Miłosz M"),
    ("Karolina K", "Nadia N"),
    ("Karolina K", "Kacper K"),
    ("Bartosz B", "Magdalena M"),
    ("Bartosz B", "Mikołaj M"),
    ("Bartosz B", "Wiktoria W"),
    ("Bartosz B", "Piotr P"),
    ("Bartosz B", "Katarzyna K"),
    ("Kamila K", "Jan J"),
    ("Kamila K", "Natalia N"),
    ("Kamila K", "Bartosz B"),
    ("Kamila K", "Sandra S"),
    ("Kamila K", "Dawid D"),
    ("Dawid D", "Emilia E"),
    ("Dawid D", "Szymon S"),
    ("Dawid D", "Karolina K"),
    ("Dawid D", "Oskar O"),
    ("Dawid D", "Klaudia K"),
    ("Dominika D", "Paweł P"),
    ("Dominika D", "Wiktoria W"),
    ("Dominika D", "Igor I"),
    ("Dominika D", "Anna A"),
    ("Dominika D", "Kamila K"),
    ("Patryk P", "Dariusz D"),
    ("Patryk P", "Magdalena M"),
    ("Patryk P", "Przemysław P"),
    ("Patryk P", "Aleksandra A"),
    ("Patryk P", "Mikołaj M"),
    ("Julia J", "Justyna J"),
    ("Julia J", "Marcin M"),
    ("Julia J", "Natalia N"),
    ("Julia J", "Krzysztof K"),
    ("Julia J", "Ewa E"),
    ("Mikołaj M", "Łukasz Ł"),
    ("Mikołaj M", "Martyna M"),
    ("Mikołaj M", "Michał M"),
    ("Mikołaj M", "Klaudia K"),
    ("Mikołaj M", "Mateusz M"),
    ("Ola O", "Julia J"),
    ("Ola O", "Piotr P"),
    ("Ola O", "Olga O"),
    ("Ola O", "Kacper K"),
    ("Ola O", "Magdalena M"),
    ("Maciej M", "Mikołaj M"),
    ("Maciej M", "Justyna J"),
    ("Maciej M", "Katarzyna K"),
    ("Maciej M", "Krzysztof K"),
```

```
    ("Maciej M", "Ewa E"),
    ("Nikola N", "Łukasz Ł"),
    ("Nikola N", "Klaudia K"),
    ("Nikola N", "Mateusz M"),
    ("Nikola N", "Julia J"),
    ("Nikola N", "Ola O"),
    ("Zuzanna Z", "Klaudia K"),
    ("Zuzanna Z", "Mateusz M"),
    ("Zuzanna Z", "Piotr P"),
    ("Zuzanna Z", "Wiktoria W"),
    ("Zuzanna Z", "Igor I"),
    ("Łukasz Ł", "Anna A"),
    ("Łukasz Ł", "Kamila K"),
    ("Łukasz Ł", "Dariusz D"),
    ("Łukasz Ł", "Magdalena M"),
    ("Łukasz Ł", "Przemysław P"),
    ("Klaudia K", "Aleksandra A"),
    ("Klaudia K", "Mikołaj M"),
    ("Klaudia K", "Justyna J"),
    ("Klaudia K", "Marcin M"),
    ("Klaudia K", "Natalia N"),
    ("Rafał R", "Krzysztof K"),
    ("Rafał R", "Ewa E"),
    ("Rafał R", "Łukasz Ł"),
    ("Rafał R", "Martyna M"),
    ("Rafał R", "Michał M"),
    ("Kornelia K", "Klaudia K"),
    ("Kornelia K", "Mateusz M"),
    ("Kornelia K", "Julia J"),
    ("Kornelia K", "Piotr P"),
    ("Kornelia K", "Olga O"),
    ("Arkadiusz A", "Natalia N"),
    ("Arkadiusz A", "Paulina P"),
    ("Arkadiusz A", "Wiktoria W"),
    ("Arkadiusz A", "Marek M"),
    ("Arkadiusz A", "Gabriel G"),
    ("Marcelina M", "Julia J"),
    ("Marcelina M", "Dominik D"),
    ("Marcelina M", "Weronika W"),
    ("Marcelina M", "Miłosz M"),
    ("Marcelina M", "Nadia N"),
    ("Marcelina M", "Kacper K"),
    ("Igor I", "Magdalena M"),
    ("Igor I", "Przemysław P"),
    ("Igor I", "Aleksandra A"),
    ("Igor I", "Mikołaj M"),
    ("Igor I", "Marcin M"),
    ("Klaudiusz K", "Natalia N"),
    ("Klaudiusz K", "Krzysztof K"),
    ("Klaudiusz K", "Ewa E"),
    ("Klaudiusz K", "Łukasz Ł"),
    ("Klaudiusz K", "Martyna M"),
    ("Martyna M", "Dominika D"),
    ("Martyna M", "Patryk P"),
    ("Martyna M", "Julia J"),
    ("Martyna M", "Natalia N"),
    ("Martyna M", "Krzysztof K"),
]

# Dodawanie krawędzi do grafu
G.add_edges_from(edges)

# Obliczanie PageRank
pagerank = nx.pagerank(G)

# Rysowanie wizualizacji sieci społecznościowej z PageRank jako wielkość węzłów
plt.figure(figsize=(10, 6))
pos = nx.spring_layout(G)  # Układ wiosenny
nx.draw(G, pos, node_size=[v * 100000 for v in pagerank.values()], with_labels=True, node_color='skyblue', font
plt.title('Wizualizacja sieci społecznościowej z PageRank')
plt.show()
```



In [18]:
```
pagerank.describe()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_15984\3313153815.py in <module>
----> 1 pagerank.describe()

NameError: name 'pagerank' is not defined
```

In [ ]:

Loading [MathJax]/extensions/Safe.js