Comportamiento del “Fondo Comunidad Activa (8% FNDR)”. Comuna de Peñaflor. Periodo 2024

In [2]:
```
import numpy as np
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import seaborn as sns
```

1. Introducción

El objetivo de este informe es analizar la distribución de proyectos financiados por el Fondo Comunidad Activa (8% FNDR), durante el año 2024. Este análisis se centra en la distribución de proyectos por ámbito, de la comuna de Peñaflor, dentro de la provincia de Talagante.

Carga de datos

La base llamada **fondo_concursable** contiene los datos de interés para el ánalisis central. La cual se obtiene en el siguiente link: https://comunidadactiva.gobiernosantiago.cl/wp-content/uploads/2023/07/Provincia-Talagante.pdf

In [6]:
```
# Lee el archivo excel y almacena en un diccionario de DataFrames
df = pd.read_excel('Fondo_Concursable.xlsx')

# Mostrar la primera fila del DataFrame concatenado
df.head(1)
```

Out[6]:

	N°	Alcance	Puntaje	Tipologia	Código	Nombre Proyecto	Institución	Tipo Organizacion	Rut
0	1	Comunal	100	Cultura	CL-00108-24	Taller de Teatro para Personas Mayores "Latido...	Fundación Educa y Colabora	FUNDACIÓN	6517

Descripción de los datos

El archivo Fondo_Concursable.xlsx contiene información detallada sobre los proyectos presentados al fondo concursable. La hoja tiene las siguientes columnas:

- **N°**: Número de identificación del proyecto.
- **Alcance**: Indica el alcance del proyecto (e.g., Comunal).
- **Puntaje**: Puntaje obtenido en el concurso, que varía entre 0 y 100.
- **Tipologia**: Tipo de proyecto (e.g., Cultura, Deporte, Seguridad, Social, Medio ambiente).

- **Código**: Código único identificador del proyecto.
- **Nombre Proyecto**: Nombre descriptivo del proyecto.
- **Institución**: Nombre de la institución que presenta el proyecto.
- **Tipo Organizacion**: Tipo de organización (e.g., Fundación, Club Deportivo, Junta de Vecinos, etc.).
- **Rut Inst.**: RUT de la institución.
- **Monto solicitado**: Monto económico solicitado para el proyecto.
- **Provincia**: Provincia asociada al proyecto.
- **Comuna**: Comuna asociada a la organización que presentó el proyecto.
- **Estado**: Resultado final del proyecto (e.g., Seleccionado, No admisible).

Análisis descriptivo estadístico

In [9]:
```
# 2. Obtener información general del DataFrame
print("\nInformación general del DataFrame:")
print(df.info())

# 3. Estadísticas descriptivas para columnas numéricas
print("\nEstadísticas descriptivas para columnas numéricas:")

descripcion = df.describe().round()

# Formatear la columna "Monto solicitado" para eliminar decimales y notación exp
descripcion['Monto solicitado'] = descripcion['Monto solicitado'].apply(lambda x

# Mostrar el resultado
descripcion
```

```
Información general del DataFrame:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 348 entries, 0 to 347
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   N°                 348 non-null    int64 
 1   Alcance            348 non-null    object
 2   Puntaje            348 non-null    int64 
 3   Tipologia          348 non-null    object
 4   Código             348 non-null    object
 5   Nombre Proyecto    348 non-null    object
 6   Institución        348 non-null    object
 7   Tipo Organizacion  348 non-null    object
 8   Rut Inst.          348 non-null    object
 9   Monto solicitado   348 non-null    int64 
 10  Provincia          348 non-null    object
 11  Comuna             348 non-null    object
 12  Estado             348 non-null    object
 13  Unnamed: 13        4 non-null      object
dtypes: int64(3), object(11)
memory usage: 38.2+ KB
None

Estadísticas descriptivas para columnas numéricas:
```

Out[9]:

	N°	Puntaje	Monto solicitado
count	348.0	348.0	348
mean	109.0	65.0	5,800,769
std	76.0	40.0	3,415,849
min	1.0	0.0	659,000
25%	44.0	36.0	2,174,852
50%	88.0	85.0	5,617,162
75%	174.0	95.0	9,536,285
max	261.0	100.0	10,000,000

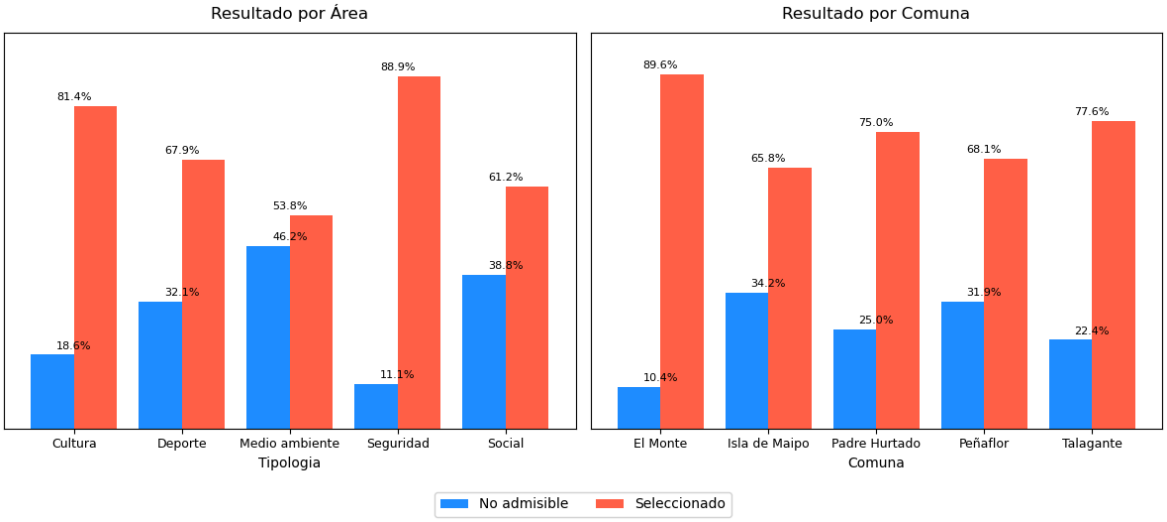
El análisis estadístico descríptivo de las variables: Puntaje y Monto Solicitado resalta que no existen valores nulos o fuera de rango. La base es pequeña y contiene datos bien precisos, donde se destaca el monto minimo de dinero solicitado, la suma de 659.000 de pesos y el monto máximo, de 10.000.000 de pesos.

Visualización de Los Resultados del Fondo.

In [160…

```
# Crear una figura con dos subplots (1 fila, 2 columnas)
fig, axes = plt.subplots(1, 2, figsize=(12, 5))  # Reducir el tamaño de la figur

# Graficar el primer gráfico (Tipologia)
bars_tipologia = porcentajes_cruzados_tipologia.plot(kind='bar', color=['dodgerb
axes[0].set_title('Resultado por Área', fontsize=12, pad=10)  # Título más grana
axes[0].tick_params(axis='x', rotation=0, labelsize=9)  # Reducir el tamaño de l
axes[0].set_yticks([])  # Eliminar las marcas del eje y
axes[0].set_yticklabels([])  # Eliminar las etiquetas del eje y
axes[0].set_ylim(0, 100)  # Limitar el rango del eje y para que las barras no se

# Añadir etiquetas de porcentaje sobre las barras (Tipologia)
for i, (colname, data) in enumerate(porcentajes_cruzados_tipologia.items()):
    for j, value in enumerate(data):
        axes[0].annotate(f'{value:.1f}%', xy=(j, value + 1), ha='center', va='bo

# Graficar el segundo gráfico (Comuna)
bars_comuna = porcentajes_cruzados_comuna.plot(kind='bar', color=['dodgerblue',
axes[1].set_title('Resultado por Comuna', fontsize=12, pad=10)  # Título más gra
axes[1].tick_params(axis='x', rotation=0, labelsize=9)  # Reducir el tamaño de l
axes[1].set_yticks([])  # Eliminar las marcas del eje y
axes[1].set_yticklabels([])  # Eliminar las etiquetas del eje y
axes[1].set_ylim(0, 100)  # Limitar el rango del eje y para que las barras no se

# Añadir etiquetas de porcentaje sobre las barras (Comuna)
for i, (colname, data) in enumerate(porcentajes_cruzados_comuna.items()):
    for j, value in enumerate(data):
        axes[1].annotate(f'{value:.1f}%', xy=(j, value + 1), ha='center', va='bo

# Crear una leyenda manual y compartirla entre los gráficos
handles, labels = axes[0].get_legend_handles_labels()
fig.legend(handles, labels, loc='upper center', bbox_to_anchor=(0.5, 0), ncol=2,

# Ajustar el espacio entre los subplots
```

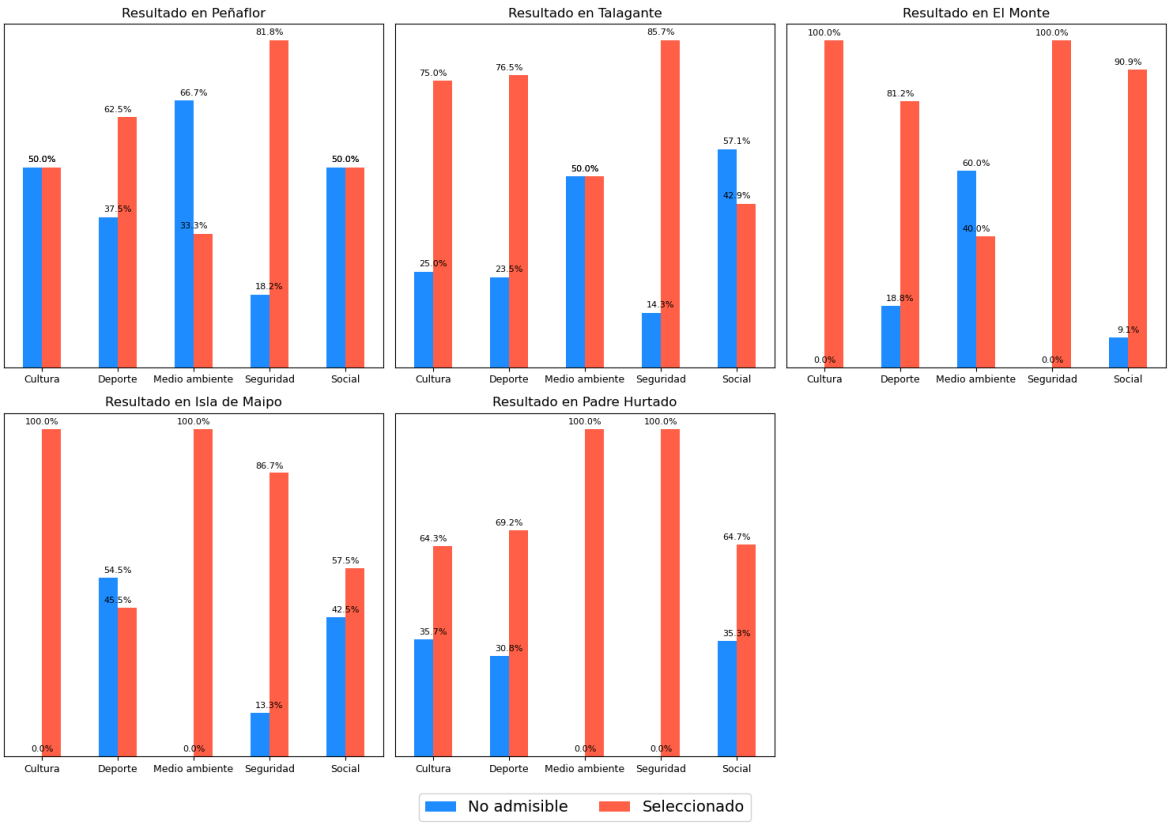
```
plt.tight_layout()

# Mostrar la figura
plt.show()
```



In [132…

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

# Lista de comunas
comunas = ['Peñaflor', 'Talagante', 'El Monte', 'Isla de Maipo', 'Padre Hurtado'

# Crear una figura con 5 subplots (uno por comuna)
fig, axes = plt.subplots(2, 3, figsize=(15, 10))  # 2 filas, 3 columnas (el últi
axes = axes.flatten()  # Aplanar la matriz de subplots para facilitar el acceso

# Iterar sobre cada comuna y crear un gráfico
for i, comuna in enumerate(comunas):
    # Filtrar el DataFrame por comuna
    df_comuna = df[df['Comuna'] == comuna]

    # Crear una tabla cruzada para la comuna y el estado
    frecuencias_cruzadas = pd.crosstab(df_comuna['Tipologia'], df_comuna['Estado

    # Calcular los porcentajes
    porcentajes_cruzados = frecuencias_cruzadas.div(frecuencias_cruzadas.sum(axi

    # Graficar el gráfico de barras
    bars = porcentajes_cruzados.plot(kind='bar', color=['dodgerblue', 'tomato'],
    axes[i].set_title(f'Resultado en {comuna}')  # Título del gráfico (sin la pa
    axes[i].tick_params(axis='x', rotation=0, labelsize=9)  # Reducir el tamaño
    axes[i].set_yticks([])  # Eliminar las marcas del eje y
    axes[i].set_yticklabels([])  # Eliminar las etiquetas del eje y
    axes[i].set_xlabel('')  # Eliminar la etiqueta del eje x (si existía)

    # Añadir etiquetas de porcentaje sobre las barras
    for j, (colname, data) in enumerate(porcentajes_cruzados.items()):
        for k, value in enumerate(data):
            axes[i].annotate(f'{value:.1f}%', xy=(k, value + 1), ha='center', va

# Ocultar el sexto subplot (posición [1, 2])
axes[-1].axis('off')

# Crear una leyenda manual y compartirla entre los gráficos
```

```
handles, labels = bars.get_legend_handles_labels()
fig.legend(handles, labels, loc='upper center', bbox_to_anchor=(0.5, 0), ncol=2,

# Ajustar el espacio entre los subplots
plt.tight_layout()

# Mostrar la figura
plt.show()
```



En resultados por ámbito destaca que el **peor** rendimiento se da en **Medio Ambiente** donde la diferencia entre seleccionados y no admisibles es poca 7.6% y el **mejor** es en **Seguridad** con una diferencia de 78.8%.

En resultados por comuna destaca **El Monte** con una diferencia de 79.2% entre los proyectos seleccionados y los no seleccionados. Las otras comunas tienen un comportamiento casi igual.

La comuna de **Peñaflor** (tercer gráfico) muestra un comportamiento no muy alentador. En medio ambiente son más los proyectos rechazados que aprobados y en cultura y social solo la mitad de los proyectos son aprobados. Seguridad y deporte son los únicos que tienen más proyectos ganados que rechazados.

In [14]:
```
# Crear una tabla de contingencia entre Tipologia y Estado en Peñaflor
conteo_penaflor = pd.crosstab(df_penaflor['Tipologia'], df_penaflor['Estado'])

# Mostrar el resultado
print('Comuna de Peñaflor')
conteo_penaflor
```

```
Comuna de Peñaflor
```

Out[14]:

Estado	No admisible	Seleccionado
Tipologia		
Cultura	1	1
Deporte	9	15
Medio ambiente	2	1
Seguridad	6	27
Social	5	5

In [182…

```
import pandas as pd

# Crear una tabla cruzada (pivot table) para el Estado por Comuna (solo conteo)
tabla_cruzada_estado = df.pivot_table(
    values='N°',                # Usar una columna numérica para contar
    index='Comuna',             # Filas (comunas)
    columns='Estado',           # Columnas (estados)
    aggfunc='count',            # Función de agregación (contar)
    fill_value=0                # Rellenar valores faltantes con 0
)

# Calcular el total general (total de proyectos)
total_general = tabla_cruzada_estado.sum().sum()

# Agregar totales a la tabla de conteo
tabla_cruzada_estado['Total Comuna'] = tabla_cruzada_estado.sum(axis=1)  # Total
tabla_cruzada_estado.loc['Total Estado'] = tabla_cruzada_estado.sum(axis=0)  # T

# Calcular la tabla de porcentajes
tabla_cruzada_porcentajes = (tabla_cruzada_estado / total_general) * 100
tabla_cruzada_porcentajes = tabla_cruzada_porcentajes.map(lambda y: f"{y:.2f}%")

# Renombrar las columnas de la tabla de porcentajes para evitar conflictos
tabla_cruzada_porcentajes.columns = [f"{col} (%)" for col in tabla_cruzada_porce

# Combinar las dos tablas horizontalmente
tabla_combinada = pd.concat([tabla_cruzada_estado, tabla_cruzada_porcentajes], a

# Mostrar la tabla combinada
print("Tabla cruzada del estado de los proyectos por comuna (conteo y porcentaje
tabla_combinada
```

```
Tabla cruzada del estado de los proyectos por comuna (conteo y porcentajes)
```

Out[182…

Comuna	No admisible	Seleccionado	Total Comuna	No admisible (%)	Seleccionado (%)	Total Comuna (%)
El Monte	7	60	67	2.01%	17.24%	19.25%
Isla de Maipo	25	48	73	7.18%	13.79%	20.98%
Padre Hurtado	15	45	60	4.31%	12.93%	17.24%
Peñaflor	23	49	72	6.61%	14.08%	20.69%
Talagante	17	59	76	4.89%	16.95%	21.84%
Total Estado	87	261	348	25.00%	75.00%	100.00%

In [184…
```
# Contar los datos de la variable 'Estado'
total_estados = df['Estado'].value_counts()

# Mostrar el resultado
total_estados
```

Out[184…
```
Estado
Seleccionado    261
No admisible     87
Name: count, dtype: int64
```

Proyectos Seleccionados

Se analizará y visualizará solo los proyectos SELECCIONADOS entre el total de los presentados.

In [49]:
```
# Generar solo el Estado "Seleccionado"
estado_seleccionado = 'Seleccionado'

# Filtrar el DataFrame por el estado seleccionado
df_filtrado = df[df['Estado'] == estado_seleccionado]
```

In [51]:
```
# Crear una tabla cruzada (pivot table)
tabla_cruzada = df_filtrado.pivot_table(
    values='Monto solicitado',  # Valores que se sumarán
    index='Comuna',             # Filas (comunas)
    columns='Tipologia',        # Columnas (tipologías)
    aggfunc='sum',              # Función de agregación (suma)
    fill_value=0                # Rellenar valores faltantes con 0
)

# Agregar una columna con el total por fila (total por comuna)
tabla_cruzada['Total Comuna'] = tabla_cruzada.sum(axis=1)

# Agregar una fila con el total por columna (total por tipología)
tabla_cruzada.loc['Total Area'] = tabla_cruzada.sum(axis=0)

# Formatear los valores de la tabla cruzada (sin decimales y con separadores de
```

```
tabla_cruzada_formateada = tabla_cruzada.map(lambda y: f"{int(y):,}")

# Mostrar la tabla cruzada formateada
print("Tabla cruzada del monto solicitado por comuna y tipología:")
tabla_cruzada_formateada
```

```
Tabla cruzada del monto solicitado por comuna y tipología:
```

Out[51]:

Tipologia	Cultura	Deporte	Medio ambiente	Seguridad	Social	Total Comuna
Comuna						
El Monte	89,633,747	52,535,680	10,124,866	179,806,494	53,949,826	386,050,613
Isla de Maipo	44,652,106	9,043,816	14,326,396	119,218,566	66,169,856	253,410,740
Padre Hurtado	22,902,729	29,012,873	2,422,000	133,630,414	39,482,983	227,450,999
Peñaflor	9,665,159	69,692,495	9,748,293	239,154,475	30,857,620	359,118,042
Talagante	41,198,758	45,111,990	9,999,990	295,171,970	12,638,427	404,121,135
Total Area	208,052,499	205,396,854	46,621,545	966,981,919	203,098,712	1,630,151,529

In [21]:

```
# Filtrar el DataFrame por el estado seleccionado
df_filtrado = df[df['Estado'] == estado_seleccionado]

# Filtrar solo la Comuna de Peñaflor
df_peñaflor = df_filtrado[df_filtrado['Comuna'] == 'Peñaflor']

# Agrupar por "Comuna", "Tipo de Organización" y "Tipología", y aplicar múltiple
resultado = df_peñaflor.groupby(['Comuna', 'Tipo Organizacion', 'Tipologia']).ag
    Cantidad_Proyectos=('Rut Inst.', 'count'),  # Contar la cantidad de proyecto
    Monto_Solicitado_Total=('Monto solicitado', 'sum')  # Sumar el Monto Solicit
).reset_index()

# Ordenar por Comuna y Cantidad de Proyectos (aunque solo hay una comuna, Peñafl
resultado = resultado.sort_values(by=['Tipo Organizacion', 'Cantidad_Proyectos']

# Formatear los valores numéricos
resultado['Cantidad_Proyectos'] = resultado['Cantidad_Proyectos'].map(lambda x:
resultado['Monto_Solicitado_Total'] = resultado['Monto_Solicitado_Total'].map(la

# Configurar pandas para mostrar todas las filas
pd.set_option('display.max_rows', None)

# Mostrar el resultado
print(f"Proyectos seleccionados en Peñaflor por tipo de organización y tipología
resultado
```

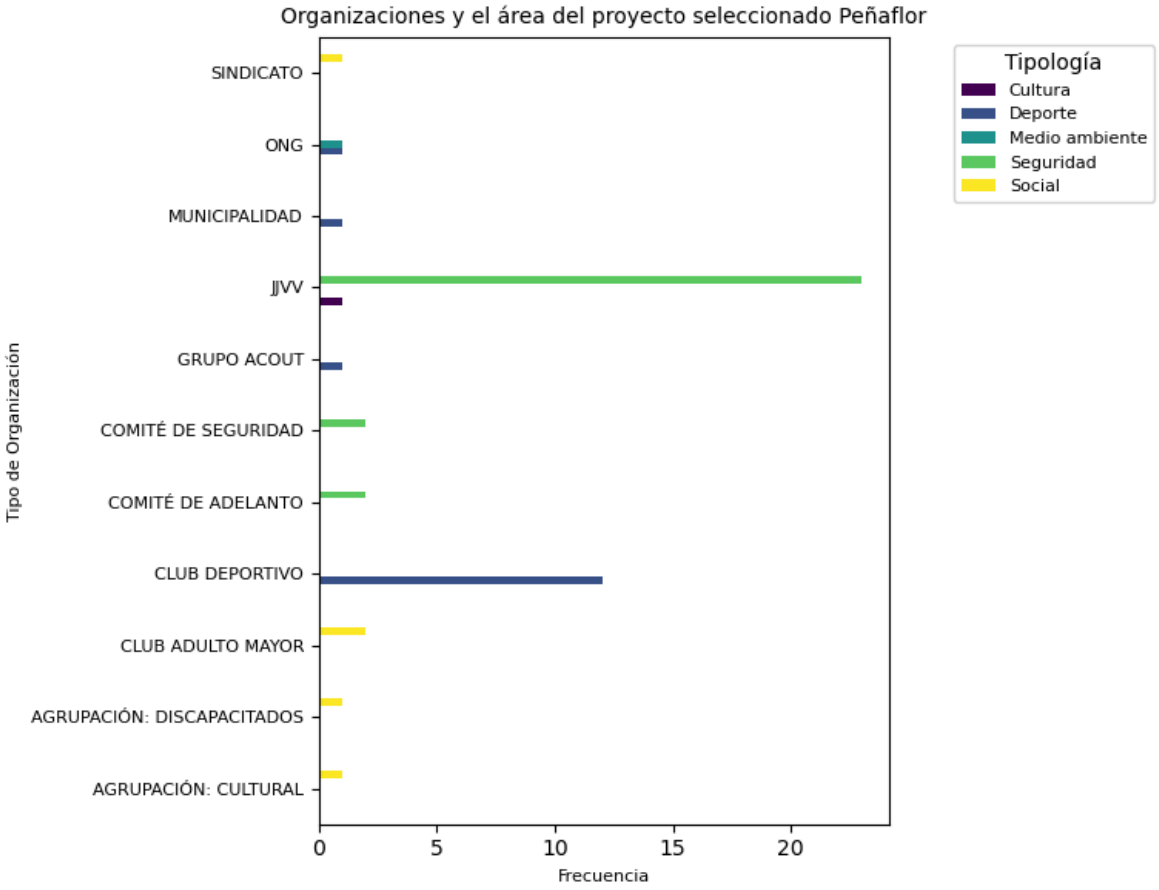
```
Proyectos seleccionados en Peñaflor por tipo de organización y tipología:
```

Out[21]:

	Comuna	Tipo Organizacion	Tipologia	Cantidad_Proyectos	Monto_Solicitado_Total
0	Peñaflor	AGRUPACIÓN: CULTURAL	Social	1	$9,835,210
1	Peñaflor	AGRUPACIÓN: DISCAPACITADOS	Social	1	$5,139,820
2	Peñaflor	CLUB ADULTO MAYOR	Social	2	$5,985,435
3	Peñaflor	CLUB DEPORTIVO	Deporte	12	$56,205,245
4	Peñaflor	COMITÉ DE ADELANTO	Seguridad	2	$17,968,792
5	Peñaflor	COMITÉ DE SEGURIDAD	Seguridad	2	$18,285,511
6	Peñaflor	GRUPO ACOUT	Deporte	1	$2,179,000
8	Peñaflor	JJVV	Seguridad	23	$202,900,172
7	Peñaflor	JJVV	Cultura	1	$9,665,159
9	Peñaflor	MUNICIPALIDAD	Deporte	1	$1,917,250
10	Peñaflor	ONG	Deporte	1	$9,391,000
11	Peñaflor	ONG	Medio ambiente	1	$9,748,293
12	Peñaflor	SINDICATO	Social	1	$9,897,155

In [110…

```
comuna_seleccionada = 'Peñaflor'  # lo que deseas filtrar

# Filtrar el DataFrame por el estado seleccionado
df_penaflor = df_filtrado[df_filtrado['Comuna'] == comuna_seleccionada]

# Crear una tabla de contingencia entre Tipo de Organización y Tipología
tabla_contingencia = pd.crosstab(df_penaflor['Tipo Organizacion'], df_penaflor['

# Crear el gráfico de barras horizontales
tabla_contingencia.plot(kind='barh', figsize=(8, 6), colormap='viridis')

# Personalizar el gráfico
plt.title('Organizaciones y el área del proyecto seleccionado Peñaflor', fontsiz
plt.xlabel('Frecuencia', fontsize=8)
plt.ylabel('Tipo de Organización', fontsize=8)
plt.yticks(fontsize=8)
plt.legend(title='Tipología', fontsize=8, bbox_to_anchor=(1.1, 1), loc='upper le

# Mostrar el gráfico
plt.tight_layout()
plt.show()
```


Organizaciones y el área del proyecto seleccionado Peñaflor
Tipología
Cultura
Deporte
Medio ambiente
Seguridad
Social
Tipo de Organización
SINDICATO
ONG
MUNICIPALIDAD
JJVV
GRUPO ACOUT
COMITÉ DE SEGURIDAD
COMITÉ DE ADELANTO
CLUB DEPORTIVO
CLUB ADULTO MAYOR
AGRUPACIÓN: DISCAPACITADOS
AGRUPACIÓN: CULTURAL
0
5
10
15
20
Frecuencia