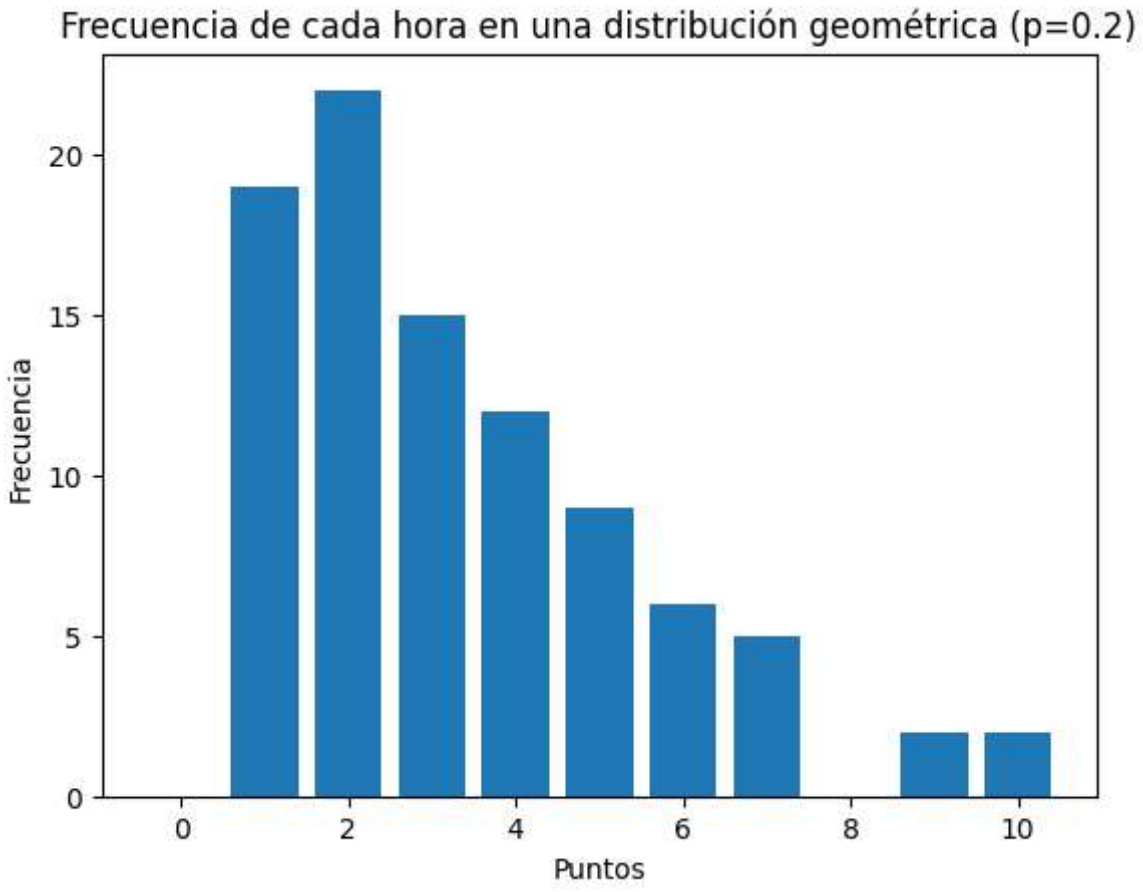
Task 1

In [31]:
```
import numpy as np
import matplotlib.pyplot as plt

# Definir la probabilidad de éxito (p)
p = 0.2

# Generar una muestra, en este caso definimos 100 imaginando que 100 personas
# llegaron a una tienda al día y el promedio diario es el encontrado con la distrib
muestra = np.random.geometric(p, size=100)

# Filtrar los valores para que estén en el rango deseado (0 a 10). Esto lo hacemos
# pues solo deseamos analizar lo que ocurre en las diez horas de apretura de la tie
muestra_filtrada = muestra[(muestra >= 0) & (muestra <= 10)]

# Calcular la frecuencia de cada hora
frecuencia = [np.sum(muestra_filtrada == i) for i in range(11)]

# Gráfico de barras
puntos = list(range(11))
plt.bar(puntos, frecuencia)
plt.xlabel('Puntos')
plt.ylabel('Frecuencia')
plt.title('Frecuencia de cada hora en una distribución geométrica (p=0.2)')
plt.show()
```



In [32]:
```
# Calculamos el promedio de llegadas por hora
promedio_llegadas_por_hora = round(np.mean(frecuencia))

# Calculamos el tiempo promedio entre llegadas (en minutos)
tiempo_promedio_entre_llegadas = round((1 / promedio_llegadas_por_hora)*60)

print("Tiempo promedio entre llegadas (minutos):", tiempo_promedio_entre_llegadas)
```

```
Tiempo promedio entre llegadas (minutos): 8
```

In [33]:
```
# Calcular el valor esperado (media) en una distribución geométrica
valor_esperado = 1 / p
print("Valor Esperado (Media):", valor_esperado)
```

```
Valor Esperado (Media): 5.0
```

¿Qué modela la distribución geométrica en este escenario?

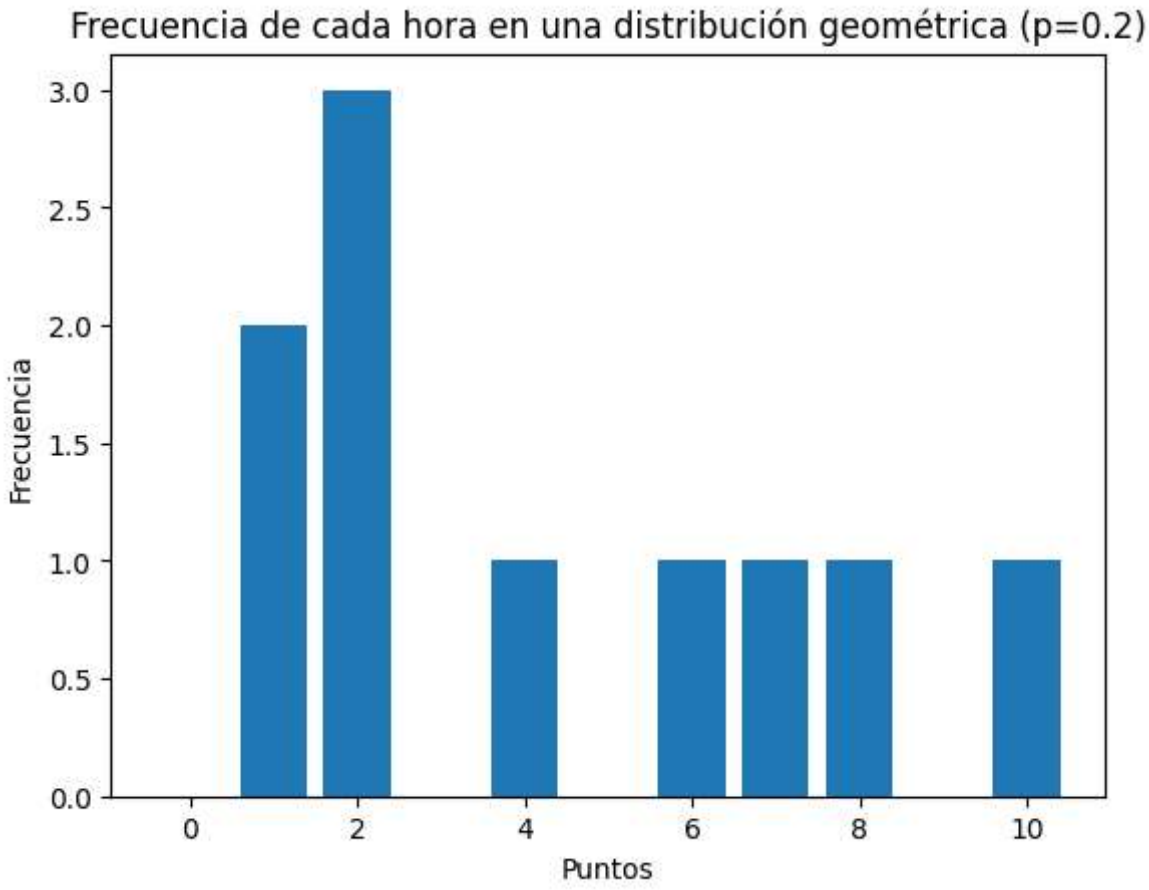
Modela el comportamiento de llegada de 100 personas en un periodo de 10 horas. Asumiendo que el comportamiento se da con una distribución geométrica y que todos los casos se dieron en un día, de esta manera podemos encontrar la frecuencia de llegada y definir un promedio de llegada por cada cliente.

¿Cómo se compara el tiempo promedio entre llegadas con el valor esperado de la distribución geométrica?

En este caso la distribución geométrica coloca sobre un punto en la recta la llegada de una persona a la tienda, como podemos ver en la gráfica el comportamiento dado la función P(X = x) = (1 - p)^(x-1) * p causa que las llegadas disminuyan conforme pasa el tiempo. El valor esperado nos dice cuántos eventos deben pasar antes que un éxito ocurra. Para el primer valor la posibilidad de éxito es de 1/p, es decie 1-0.2 lo cual nos indica que deben pasar 5 eventos para que uno sea exitoso, es decir que una persona llegue en la hora 1. Por lo tanto podemos deducir que, si la llegada de 20 personas en la hora uno donde p es 0.2, tuvieron que pasar teóricamente 20*5 (100) personas delante de la tienda para que las 20 personas entraran a esa hora.

Ahora, hay otra forma de ver este escenario y es que se den 10 eventos y, en base a esos 10 eventos, analicemos el comportamiento y resultados

In [44]:
```
import numpy as np
import matplotlib.pyplot as plt

# Definir la probabilidad de éxito (p)
p = 0.2

# Generar una muestra, en este caso definimos 10 imaginando que se da el evento una
muestra = np.random.geometric(p, size=10)

# Calcular la frecuencia de cada hora
frecuencia = [np.sum(muestra == i) for i in range(11)]

# Gráfico de barras
puntos = list(range(11))
plt.bar(puntos, frecuencia)
plt.xlabel('Puntos')
plt.ylabel('Frecuencia')
plt.title('Frecuencia de cada hora en una distribución geométrica (p=0.2)')
plt.show()
```



In [42]:
```
# Calculamos el promedio de llegadas por hora
promedio_llegadas_por_hora = round(np.mean(frecuencia))

# Calculamos el tiempo promedio entre llegadas (en minutos)
tiempo_promedio_entre_llegadas = round((1 / promedio_llegadas_por_hora)*60)

print("Tiempo promedio entre llegadas (minutos):", tiempo_promedio_entre_llegadas)
```

```
Tiempo promedio entre llegadas (minutos): 60
```

In [43]:
```
# Calcular el valor esperado (media) en una distribución geométrica
valor_esperado = 1 / p
print("Valor Esperado (Media):", valor_esperado)
```

```
Valor Esperado (Media): 5.0
```

¿Qué modela la distribución geométrica en este escenario?

Modela el comportamiento de llegada de las personas dado 10 eventos ocurridos. Uno por cada hora.

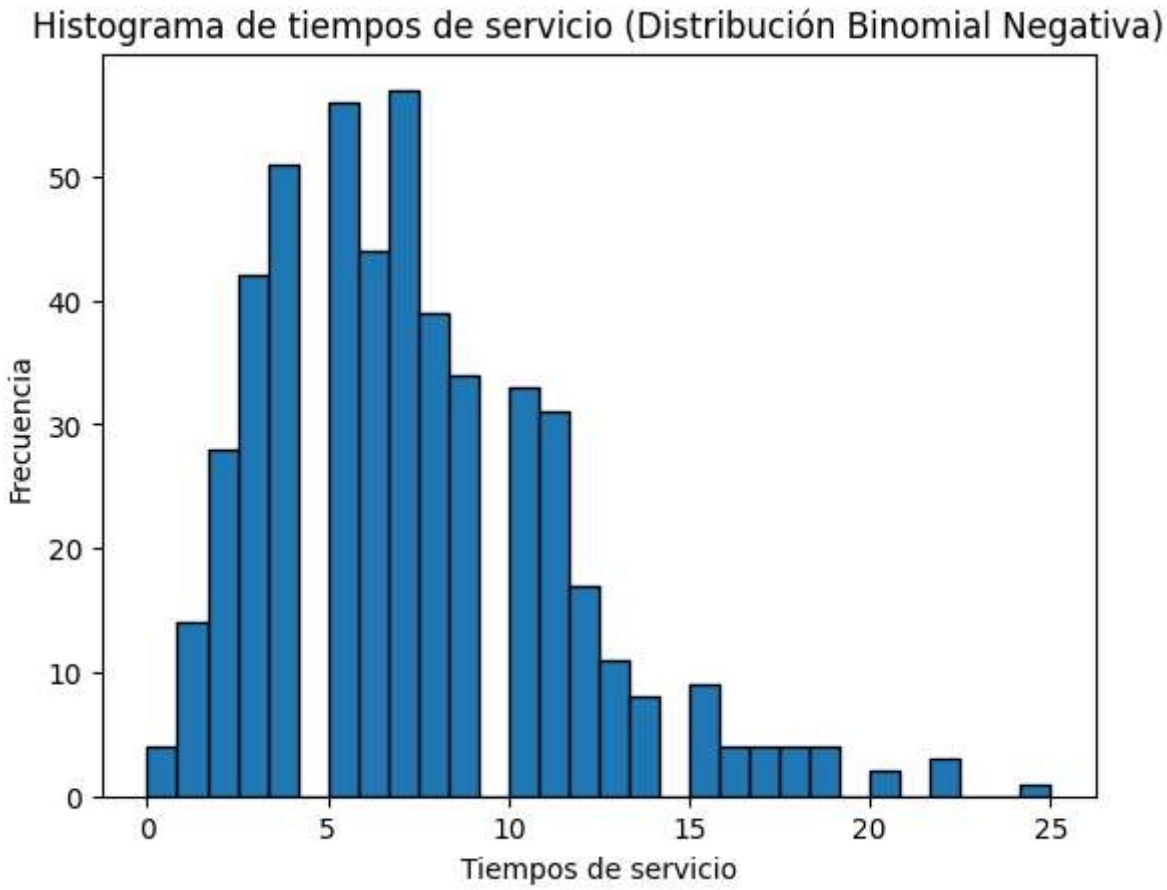
¿Cómo se compara el tiempo promedio entre llegadas con el valor esperado de la distribución geométrica?

En este caso el promedio de llegada sería 60 pues se dió un evento por hora básicamente. Por otro lado el valor esperado sigue siendo igual representando la cantidad de eventos que

deben ocurrir antes de que un evento sea exitoso en la hora correspondiente. En este caso podemos comparar los resultados obtenidos en los éxitos y los esperados.

Task 2

In [52]:
```
import numpy as np
import matplotlib.pyplot as plt

# Definir parámetros de la distribución binomial negativa
r = 5  # número de éxitos requeridos
p = 0.4  # probabilidad de éxito en cada ensayo

# Generar una muestra aleatoria de 500 tiempos de servicio
muestra_tiempos_servicio = np.random.negative_binomial(r, p, size=500)

# Calcular la media y la varianza de la muestra
media_muestra = np.mean(muestra_tiempos_servicio)
varianza_muestra = np.var(muestra_tiempos_servicio)

# Gráfico de histograma
plt.hist(muestra_tiempos_servicio, bins=30, edgecolor='black')
plt.xlabel('Tiempos de servicio')
plt.ylabel('Frecuencia')
plt.title('Histograma de tiempos de servicio (Distribución Binomial Negativa)')
plt.show()

# Imprimir la media y la varianza de la muestra
print("Media de la muestra:", media_muestra)
print("Varianza de la muestra:", varianza_muestra)
```




```
Media de la muestra: 7.274
Varianza de la muestra: 17.278924
```

¿Cómo se ajusta la distribución binomial negativa a los datos de tiempo de servicio?

Modificando los parámetros de ensayo como r y p, a partir de esto es que podemos adecuar la distribución a los datos de tiempo de servicio, claro esto depende completamente del contexto.

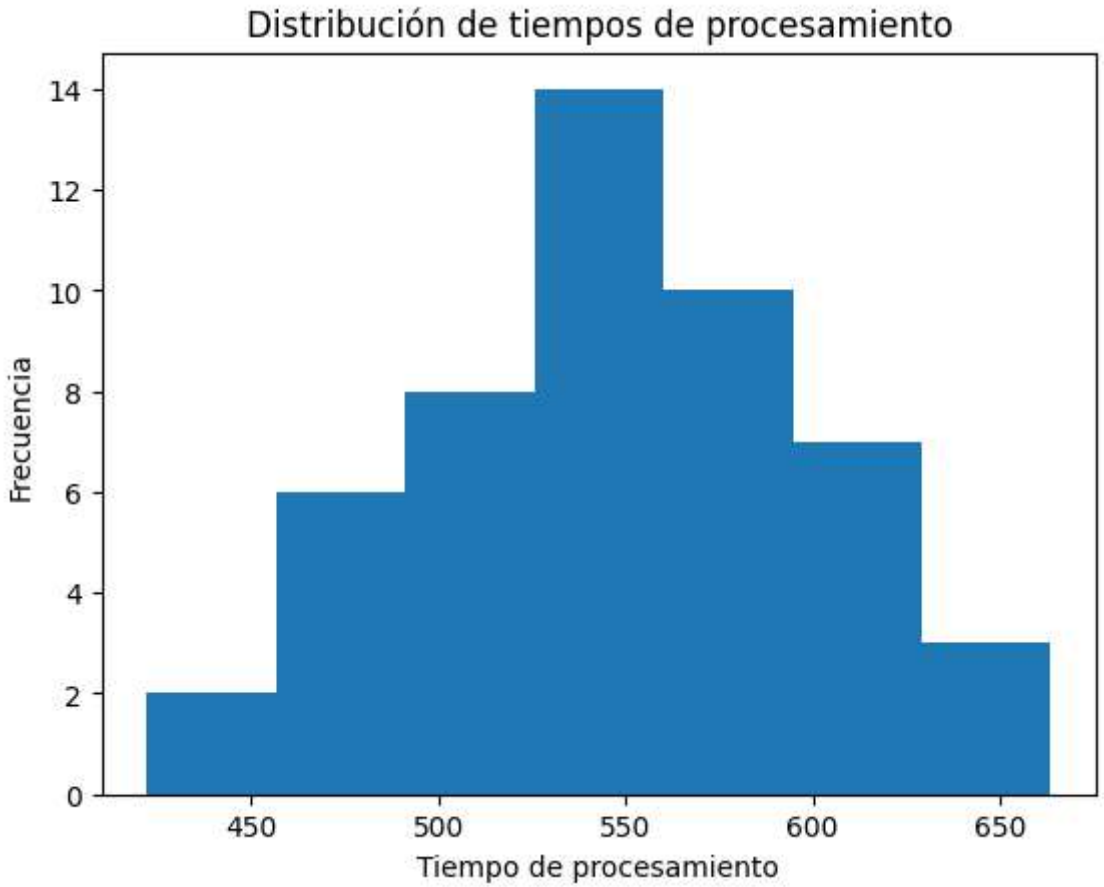
Compare la media y la varianza de los datos con los valores teóricos de la distribución binomial negativa.

Como se puede observar la media dice que el valor más frecuentado es 7 y que hay una variabilidad media de esta serie de datos.

Task 3

In [62]:
```
import numpy as np
import matplotlib.pyplot as plt

# 1. Simulación de los tiempos de procesamiento
N = 50  # total de tareas
K = 10  # total de artículos defectuosos
```

```
n = 50  # tamaño de la muestra
procesamiento_defectuoso = 15  # tiempo de procesamiento de un artículo defectuoso
procesamiento_no_defectuoso = 10  # tiempo de procesamiento de un artículo no defec
ruido = np.random.normal(0, 1, n)  # ruido aleatorio, definimos este ruido para hac
                                   # saldría una caja sin sentido.

np.random.seed(0)  # para la reproducibilidad
tiempos = np.random.hypergeometric(K, N-K, n)
tiempos = tiempos * (procesamiento_defectuoso + ruido) + (n - tiempos) * (procesami

# 2. Graficar los tiempos de procesamiento y calcular media y varianza
plt.hist(tiempos, bins='auto')
plt.title('Distribución de tiempos de procesamiento')
plt.xlabel('Tiempo de procesamiento')
plt.ylabel('Frecuencia')
plt.show()

print('Media de los tiempos de procesamiento:', np.mean(tiempos))
print('Varianza de los tiempos de procesamiento:', np.var(tiempos))
```




```
Media de los tiempos de procesamiento: 549.444121492987
Varianza de los tiempos de procesamiento: 2883.829448671045
```

¿Cómo modela la distribución hipergeométrica los tiempos de procesamiento en este escenario?

Como se puede observar en la gráfica el comportamiento es normal con gran concentración en el centro, podríamos asumir que el mayor tiempo es de los eventos defectuosos y los

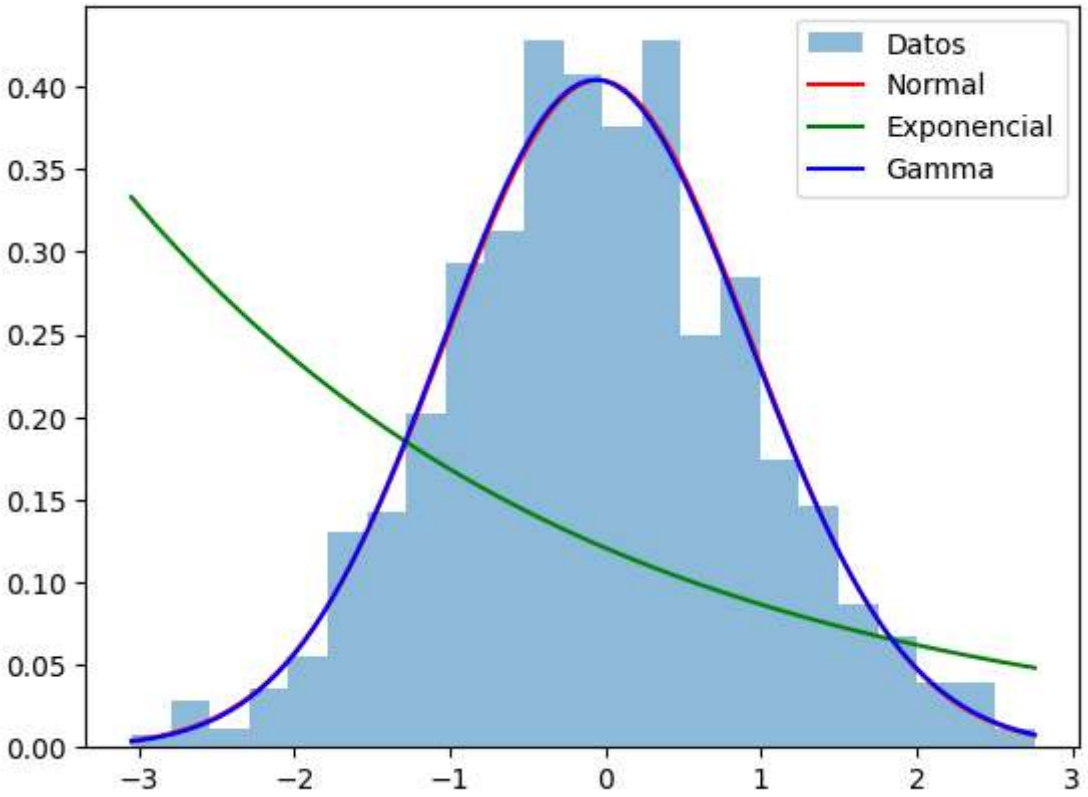
veloces por eventos normales

¿Existen patrones en los tiempos de procesamiento de los artículos defectuosos frente a los no defectuosos?

Teóricamente los defectuosos deberían tardar más que los no defectuosos

Task 4

In [63]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm, expon, gamma, kstest

# 1. Generar muestra aleatoria de 1000 puntos de datos de una distribución desconoc
np.random.seed(0)  # para reproducibilidad
datos = np.random.normal(loc=0, scale=1, size=1000) # Asumimos que es una distribuc

# 2. Ajustar varias distribuciones a los datos mediante MLE
parametros_norm = norm.fit(datos)
parametros_expon = expon.fit(datos)
parametros_gamma = gamma.fit(datos)

# 3. Trazar histogramas de los datos y las distribuciones que mejor se ajustan en e
plt.hist(datos, bins='auto', density=True, alpha=0.5, label='Datos')

x = np.linspace(min(datos), max(datos), 100)
pdf_norm = norm.pdf(x, *parametros_norm)
pdf_expon = expon.pdf(x, *parametros_expon)
pdf_gamma = gamma.pdf(x, *parametros_gamma)

plt.plot(x, pdf_norm, 'r-', label='Normal')
plt.plot(x, pdf_expon, 'g-', label='Exponencial')
plt.plot(x, pdf_gamma, 'b-', label='Gamma')
plt.legend()
plt.show()

# 5. Realizar una prueba de bondad de ajuste para cada distribución e interpretar l
ks_stat_norm, ks_p_norm = kstest(datos, 'norm', parametros_norm)
ks_stat_expon, ks_p_expon = kstest(datos, 'expon', parametros_expon)
ks_stat_gamma, ks_p_gamma = kstest(datos, 'gamma', parametros_gamma)

print(f'Prueba KS para distribución Normal: Estadística = {ks_stat_norm}, p-valor =
print(f'Prueba KS para distribución Exponencial: Estadística = {ks_stat_expon}, p-v
print(f'Prueba KS para distribución Gamma: Estadística = {ks_stat_gamma}, p-valor =
```




```
Prueba KS para distribución Normal: Estadística = 0.01903411267034605, p-valor = 0.8
547733408587939
Prueba KS para distribución Exponencial: Estadística = 0.3456502749283127, p-valor =
3.169895355050044e-107
Prueba KS para distribución Gamma: Estadística = 0.017577786324098454, p-valor = 0.9
113300349483764
```

¿Qué distribución representa mejor los datos basados en la inspección visual?

Visualmente la distribución que mejor representa los datos basados son la normal y la gamma, interesantemente las pruebas de bondad dicen que la distribución gamma tiene mayor accuracy que la distribución normal a pesar de que generamos los datos desde una distribución normal.

In [ ]: