In [50]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical
```

In [51]:

```
# 1. Carregar e Exibir Dados
file_path = 'ligue180-2014.csv'
data = pd.read_csv(file_path, delimiter=';')
```

In [52]:

```
# Exibir as primeiras linhas e informações básicas do dataset
print("Primeiras linhas do dataset:")
print(data.head())
print("\nInformações do dataset:")
print(data.info())
```

```
Primeiras linhas do dataset:
              data_atendimento         tipo_violencia violencia_familiar  \
0  2014-01-17 00:00:00.0000000  VIOLENCIA PSICOLOGICA                Sim
1  2014-01-17 00:00:00.0000000  VIOLENCIA PSICOLOGICA                Não
2  2014-01-17 00:00:00.0000000  VIOLENCIA PSICOLOGICA                Sim
3  2014-01-17 00:00:00.0000000  VIOLENCIA PATRIMONIAL                Sim
4  2014-01-01 00:00:00.0000000       VIOLENCIA FISICA                Sim

  denunciante filhos_vitima dependencia_financeira_vitima sexo_vitima  \
0      Vítima             2                           Não    Feminino
1      Vítima             2                           Sim    Feminino
2     Cônjuge             2                           Não    Feminino
3      Vítima           NaN                           NaN    Feminino
4    Amigo(a)           NaN                           NaN         NaN

  cor_vitima faixa_etaria_vitima            escolaridade_vitima  ...  \
0     Branca  entre 40 e 44 anos  Ensino Fundamental Incompleto  ...
1        NaN  entre 35 e 39 anos                            NaN  ...
2      Parda  entre 50 e 54 anos                  Ensino Médio  ...
3        NaN  entre 45 e 49 anos                            NaN  ...
4        NaN                 NaN                            NaN  ...

  cor_agressor faixa_etaria_agressor escolaridade_agressor  \
0        Parda    entre 45 e 49 anos          Ensino Médio
1          NaN    entre 50 e 54 anos                   NaN
2        Parda    entre 50 e 54 anos    Ensino Fundamental
3          NaN                   NaN                   NaN
4          NaN                   NaN                   NaN

  drogas_alcool_agressor comportamento_efeito_agressor  \
0                    Não                         Nunca
1                 Alcool                         Nunca
2        Alcool e Drogas          Na minoria das vezes
3                    NaN                           NaN
4                    NaN                           NaN

                       filhos_violencia  \
0               Presenciam a violência
1               Presenciam a violência
2  Não presenciam nem sofrem violência
3                                  NaN
4                                  NaN

                                           violencia   uf       municipio  \
0  DANO EMOCIONAL/DIMINUIÇÃO DA AUTO-ESTIMA; AMEA...  MG   BELO HORIZONTE
1  AMEAÇA; LESÃO CORPORAL LEVE; VIOLÊNCIA PATRIMO...  RJ   RIO DE JANEIRO
2  DANO EMOCIONAL/DIMINUIÇÃO DA AUTO-ESTIMA; AMEA...  PB      JOAO PESSOA
3  DANO EMOCIONAL/DIMINUIÇÃO DA AUTO-ESTIMA; DIFA...  PB   CAMPINA GRANDE
4  ESTUPRO; VIOLÊNCIA PATRIMONIAL; DANO EMOCIONAL...  NaN             NaN

    residencia
0  Zona Urbana
1  Zona Urbana
2  Zona Urbana
3   Zona Rural
4          NaN
```

```
[5 rows x 26 columns]

Informações do dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46899 entries, 0 to 46898
Data columns (total 26 columns):
 #   Column                         Non-Null Count  Dtype 
---  ------                         --------------  ----- 
 0   data_atendimento               46899 non-null  object
 1   tipo_violencia                 46899 non-null  object
 2   violencia_familiar             43088 non-null  object
 3   denunciante                    42399 non-null  object
 4   filhos_vitima                  33937 non-null  object
 5   dependencia_financeira_vitima  33480 non-null  object
 6   sexo_vitima                    41898 non-null  object
 7   cor_vitima                     33981 non-null  object
 8   faixa_etaria_vitima            37847 non-null  object
 9   escolaridade_vitima            31932 non-null  object
 10  frequenci_violencia            33519 non-null  object
 11  coabitacao_contexto            35963 non-null  object
 12  tempo_violencia_contexto       33511 non-null  object
 13  risco_contexto                 32455 non-null  object
 14  relacao_agressor_contexto      37589 non-null  object
 15  sexo_agressor                  40745 non-null  object
 16  cor_agressor                   31451 non-null  object
 17  faixa_etaria_agressor          34354 non-null  object
 18  escolaridade_agressor          28133 non-null  object
 19  drogas_alcool_agressor         33658 non-null  object
 20  comportamento_efeito_agressor  29944 non-null  object
 21  filhos_violencia               25051 non-null  object
 22  violencia                      40451 non-null  object
 23  uf                             39146 non-null  object
 24  municipio                      39146 non-null  object
 25  residencia                     36558 non-null  object
dtypes: object(26)
memory usage: 9.3+ MB
None
```

In [53]:
```
# 2. Limpeza dos Dados
# Remover linhas com valores ausentes
data_cleaned = data.dropna()
```

In [54]:
```
# Converter colunas de data/hora para timestamps
for col in data_cleaned.select_dtypes(include=['datetime64[ns]']).columns:
    data_cleaned[col] = data_cleaned[col].astype(int) / 10**9
```

In [55]:
```
# Codificar colunas categóricas
categorical_columns = data_cleaned.select_dtypes(include=['object']).columns
data_cleaned = pd.get_dummies(data_cleaned, columns=categorical_columns)
```

In [56]:
```
# Exibir estatísticas básicas do dataset limpo
print("\nEstatísticas básicas do dataset limpo:")
print(data_cleaned.describe())
```

```
Estatísticas básicas do dataset limpo:
       data_atendimento_2014-01-01 00:00:00.0000000  \
count                                         12765   
unique                                            2   
top                                           False   
freq                                          12723   

       data_atendimento_2014-01-02 00:00:00.0000000  \
count                                         12765   
unique                                            2   
top                                           False   
freq                                          12700   

       data_atendimento_2014-01-03 00:00:00.0000000  \
count                                         12765   
unique                                            2   
top                                           False   
freq                                          12707   

       data_atendimento_2014-01-04 00:00:00.0000000  \
count                                         12765   
unique                                            2   
top                                           False   
freq                                          12717   

       data_atendimento_2014-01-05 00:00:00.0000000  \
count                                         12765   
unique                                            2   
top                                           False   
freq                                          12719   

       data_atendimento_2014-01-06 00:00:00.0000000  \
count                                         12765   
unique                                            2   
top                                           False   
freq                                          12690   

       data_atendimento_2014-01-07 00:00:00.0000000  \
count                                         12765   
unique                                            2   
top                                           False   
freq                                          12702   

       data_atendimento_2014-01-08 00:00:00.0000000  \
count                                         12765   
unique                                            2   
top                                           False   
freq                                          12711   

       data_atendimento_2014-01-09 00:00:00.0000000  \
count                                         12765   
unique                                            2   
top                                           False   
freq                                          12710   

       data_atendimento_2014-01-10 00:00:00.0000000  ...  \
```

```
count                                            12765  ...
unique                                               2  ...
top                                              False  ...
freq                                             12715  ...

       municipio_VOTUPORANGA municipio_WENCESLAU BRAZ municipio_WITMARSUM  \
count                  12765                    12765               12765
unique                     2                        2                   2
top                    False                    False               False
freq                   12763                    12764               12764

       municipio_XANGRI-LA municipio_XAPURI municipio_XAXIM  \
count                12765            12765           12765
unique                   2                2               2
top                  False            False           False
freq                 12764            12764           12764

       municipio_XINGUARA municipio_ZE DOCA residencia_Zona Rural  \
count               12765             12765                 12765
unique                  2                 2                     2
top                 False             False                 False
freq                12763             12761                 11635

       residencia_Zona Urbana
count                   12765
unique                      2
top                      True
freq                    11635

[4 rows x 9988 columns]
```

In [57]:
```python
# Salvar dados limpos em um novo arquivo CSV
cleaned_file_path = 'ligue180-2014-cleaned.csv'
data_cleaned.to_csv(cleaned_file_path, index=False)
```

In [58]:
```python
# 3. Análise com Decision Tree
# Separar variáveis independentes e dependentes
X = data_cleaned.iloc[:, :-1]
y = data_cleaned.iloc[:, -1]
```

In [59]:
```python
# Codificar a variável alvo se necessário
if y.dtype == 'object':
    y = pd.factorize(y)[0]
```

In [60]:
```python
# Dividir o dataset em conjuntos de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_sta
```

In [61]:
```python
# Inicializar e treinar o classificador Decision Tree
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)
```

Out[61]:

```
▾ DecisionTreeClassifier

DecisionTreeClassifier(random_state=42)
```

In [62]:

```python
# Fazer previsões no conjunto de teste
y_pred = clf.predict(X_test)
```

In [63]:

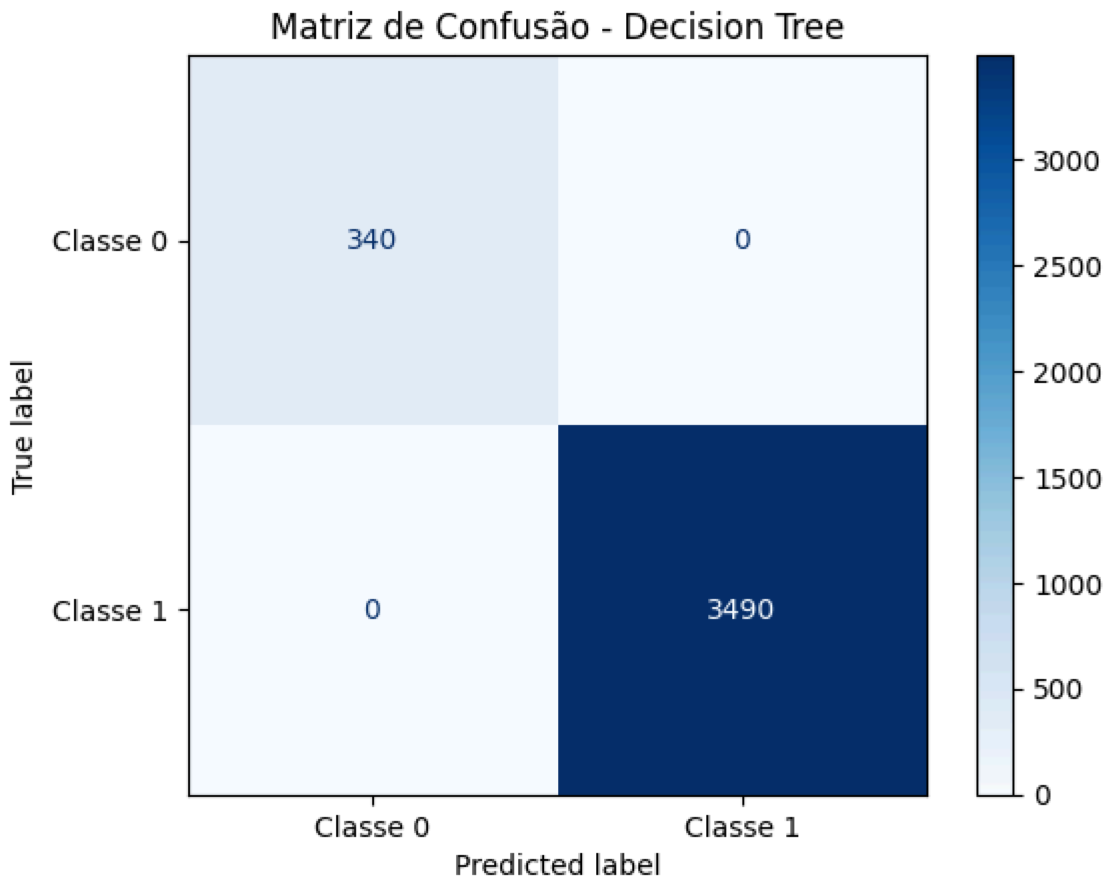
```python
# Avaliar o classificador
dt_accuracy = accuracy_score(y_test, y_pred)
dt_report = classification_report(y_test, y_pred)

print(f'\nPrecisão da Decision Tree: {dt_accuracy}')
print(f'\nRelatório de Classificação da Decision Tree:\n{dt_report}')
```

```
Precisão da Decision Tree: 1.0

Relatório de Classificação da Decision Tree:
              precision    recall  f1-score   support

       False       1.00      1.00      1.00       340
        True       1.00      1.00      1.00      3490

    accuracy                           1.00      3830
   macro avg       1.00      1.00      1.00      3830
weighted avg       1.00      1.00      1.00      3830
```

In [64]:

```python
# Plotar a Matriz de Confusão
plt.figure(figsize=(10, 7))
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Classe 0', 'Cla
disp.plot(cmap=plt.cm.Blues)
plt.title('Matriz de Confusão - Decision Tree')
plt.show()
```

```
<Figure size 1000x700 with 0 Axes>
```

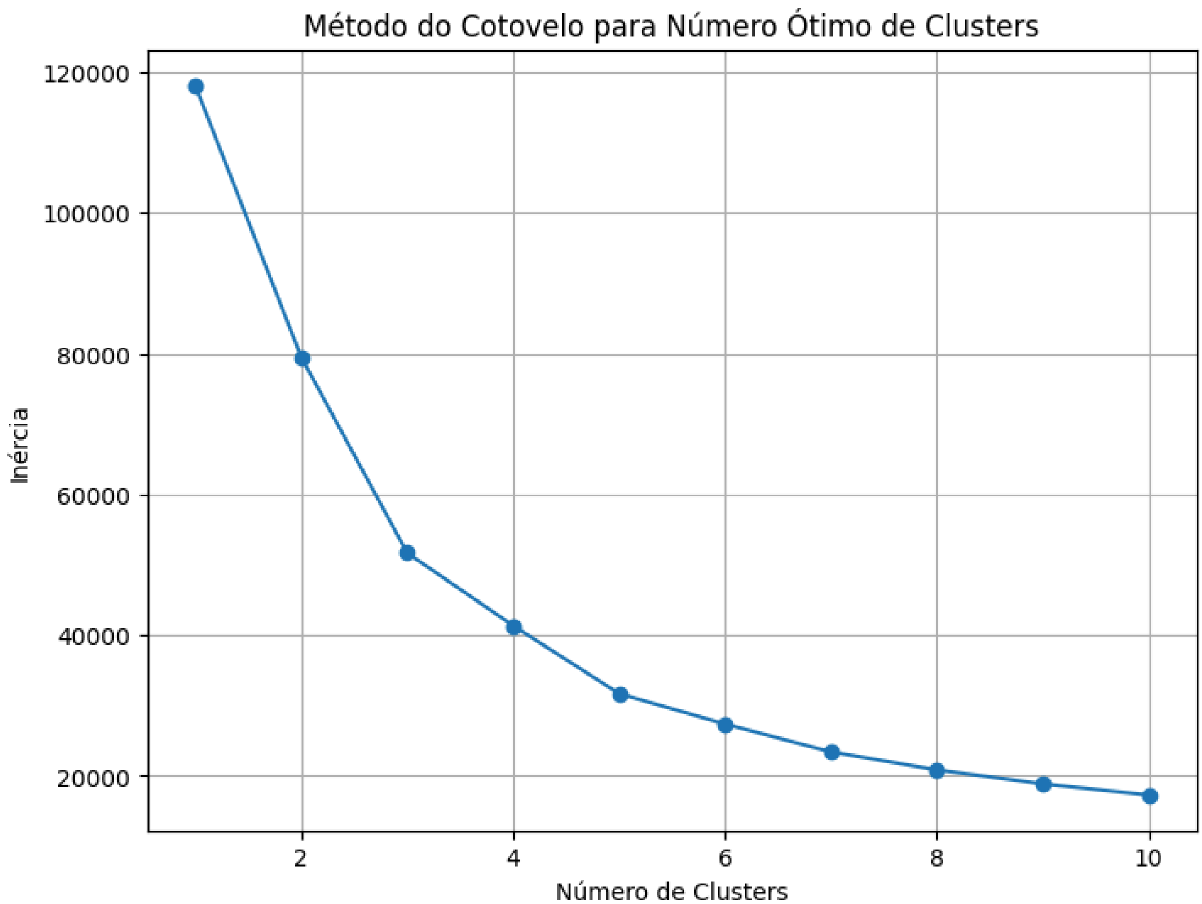


In [65]:
```
# 4. Análise com KMeans
# Padronizar os dados
scaler = StandardScaler()
data_scaled = scaler.fit_transform(X)
```

In [66]:
```
# Aplicar PCA para redução de dimensionalidade (2 componentes)
pca = PCA(n_components=2)
data_pca = pca.fit_transform(data_scaled)
```

In [67]:
```
# Determinar o número de clusters usando o método do cotovelo
inertia = []
for n in range(1, 11):
    kmeans = KMeans(n_clusters=n, random_state=42)
    kmeans.fit(data_pca)
    inertia.append(kmeans.inertia_)
```

In [68]:
```
# Plotar o método do cotovelo
plt.figure(figsize=(8, 6))
plt.plot(range(1, 11), inertia, marker='o')
plt.title('Método do Cotovelo para Número Ótimo de Clusters')
plt.xlabel('Número de Clusters')
plt.ylabel('Inércia')
plt.grid(True)
plt.show()
```

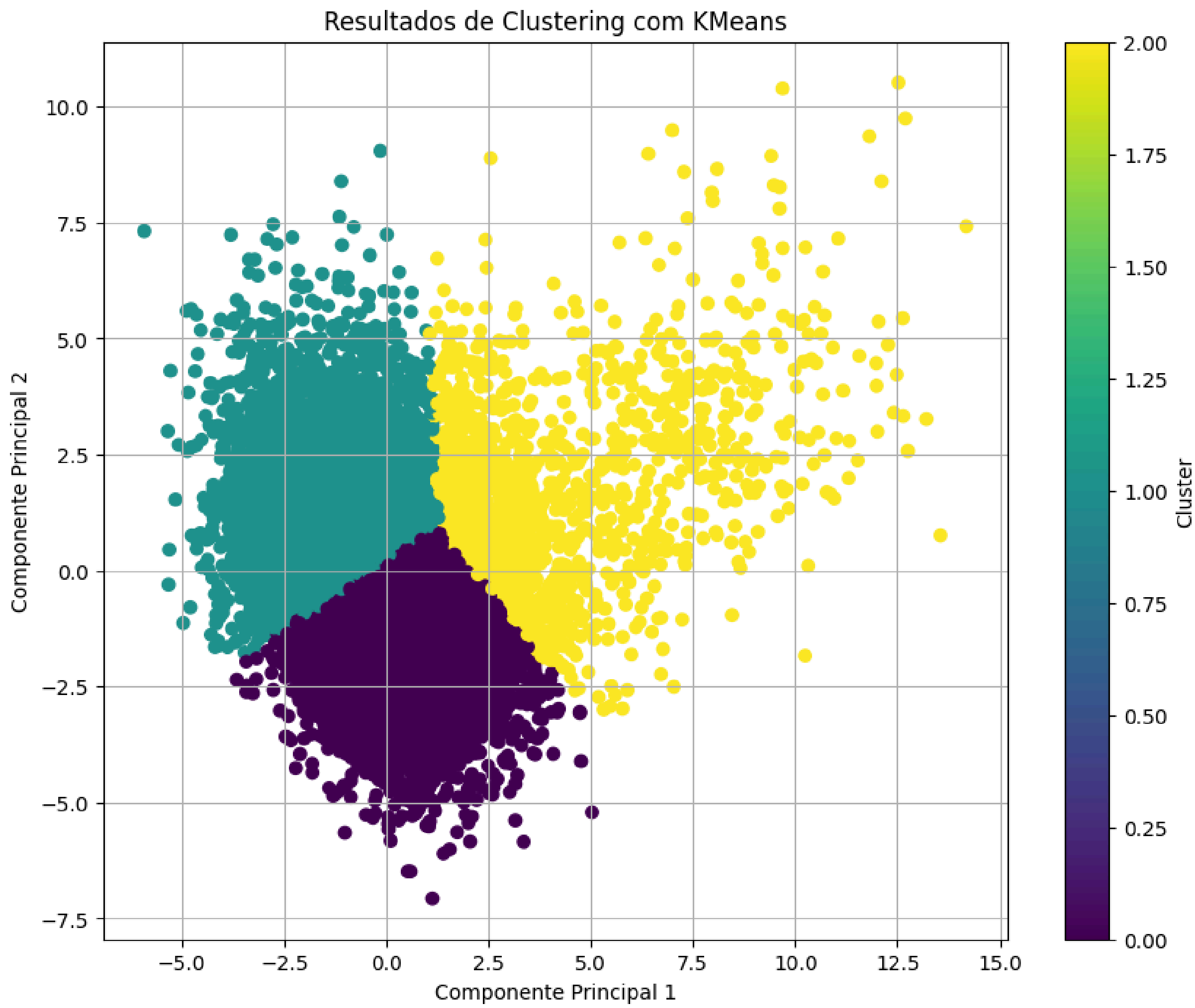


In [69]:
```
# Aplicar KMeans com o número ótimo de clusters (supondo 3)
optimal_clusters = 3
kmeans = KMeans(n_clusters=optimal_clusters, random_state=42)
clusters = kmeans.fit_predict(data_pca)
```

In [70]:
```
# Adicionar os rótulos dos clusters ao dataset original
data_cleaned['Cluster'] = clusters
```

In [71]:
```
# Salvar os dados com clusters em um novo arquivo CSV
clustered_file_path = 'ligue180-2014-clustered.csv'
data_cleaned.to_csv(clustered_file_path, index=False)
```

In [72]:
```
# Visualizar os clusters
plt.figure(figsize=(10, 8))
plt.scatter(data_pca[:, 0], data_pca[:, 1], c=clusters, cmap='viridis', marker='o')
plt.title('Resultados de Clustering com KMeans')
plt.xlabel('Componente Principal 1')
plt.ylabel('Componente Principal 2')
plt.colorbar(label='Cluster')
plt.grid(True)
plt.show()
```



In [73]:
```
# 5. Análise com Rede Neural
# Padronizar as características
X_scaled = scaler.fit_transform(X)
```

In [74]:
```
# Converter a variável alvo para categórica (one-hot encoding)
y_categorical = to_categorical(y)
```

In [75]:
```
# Dividir o dataset em conjuntos de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_categorical, test_s
```

In [76]:
```
# Inicializar a rede neural
model = Sequential()
```

In [77]:
```
# Adicionar camada de entrada e primeira camada oculta
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))
```

```
C:\Users\smour\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\l
ayers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argume
nt to a layer. When using Sequential models, prefer using an `Input(shape)` object a
s the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [78]:
```
# Adicionar segunda camada oculta
```

```
model.add(Dense(64, activation='relu'))
```

In [79]:
```
# Adicionar camada de saída
model.add(Dense(y_categorical.shape[1], activation='softmax'))
```

In [80]:
```
# Compilar o modelo
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy
```

In [81]:
```
# Treinar o modelo
history = model.fit(X_train, y_train, epochs=50, batch_size=10, verbose=1, validati
```

```
Epoch 1/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 10s 11ms/step - accuracy: 0.8777 - loss: 0.3099 - val_a
ccuracy: 0.9396 - val_loss: 0.1852
Epoch 2/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 0.9932 - loss: 0.0164 - val_ac
curacy: 0.9418 - val_loss: 0.1912
Epoch 3/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 0.9998 - loss: 0.0012 - val_ac
curacy: 0.9261 - val_loss: 0.2199
Epoch 4/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 1.0000 - loss: 2.5999e-04 - va
l_accuracy: 0.9233 - val_loss: 0.2184
Epoch 5/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 1.0000 - loss: 4.8283e-05 - va
l_accuracy: 0.9211 - val_loss: 0.2250
Epoch 6/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 1.0000 - loss: 2.6375e-05 - va
l_accuracy: 0.9217 - val_loss: 0.2322
Epoch 7/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 1.0000 - loss: 1.8924e-05 - va
l_accuracy: 0.9217 - val_loss: 0.2383
Epoch 8/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 1.0000 - loss: 1.2546e-05 - va
l_accuracy: 0.9222 - val_loss: 0.2442
Epoch 9/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 1.0000 - loss: 6.8213e-06 - va
l_accuracy: 0.9222 - val_loss: 0.2503
Epoch 10/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 1.0000 - loss: 4.9833e-06 - va
l_accuracy: 0.9194 - val_loss: 0.2565
Epoch 11/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 1.0000 - loss: 3.2132e-06 - va
l_accuracy: 0.9194 - val_loss: 0.2631
Epoch 12/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 1.0000 - loss: 2.4222e-06 - va
l_accuracy: 0.9189 - val_loss: 0.2683
Epoch 13/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 1.0000 - loss: 1.1925e-06 - va
l_accuracy: 0.9177 - val_loss: 0.2742
Epoch 14/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 1.0000 - loss: 1.0025e-06 - va
l_accuracy: 0.9155 - val_loss: 0.2798
Epoch 15/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 1.0000 - loss: 8.4175e-07 - va
l_accuracy: 0.9161 - val_loss: 0.2849
Epoch 16/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 1.0000 - loss: 4.3179e-07 - va
l_accuracy: 0.9149 - val_loss: 0.2900
Epoch 17/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 8s 11ms/step - accuracy: 1.0000 - loss: 3.0373e-07 - va
l_accuracy: 0.9133 - val_loss: 0.2951
Epoch 18/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 1.0000 - loss: 1.7132e-07 - va
l_accuracy: 0.9133 - val_loss: 0.3001
Epoch 19/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 1.0000 - loss: 1.2130e-07 - va
```

```
l_accuracy: 0.9133 - val_loss: 0.3046
Epoch 20/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 1.0000 - loss: 9.0478e-08 - va
l_accuracy: 0.9127 - val_loss: 0.3089
Epoch 21/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 1.0000 - loss: 6.2829e-08 - va
l_accuracy: 0.9127 - val_loss: 0.3130
Epoch 22/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 1.0000 - loss: 3.3022e-08 - va
l_accuracy: 0.9127 - val_loss: 0.3168
Epoch 23/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 1.0000 - loss: 2.3022e-08 - va
l_accuracy: 0.9116 - val_loss: 0.3210
Epoch 24/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 1.0000 - loss: 1.5218e-08 - va
l_accuracy: 0.9110 - val_loss: 0.3247
Epoch 25/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 1.0000 - loss: 1.2346e-08 - va
l_accuracy: 0.9105 - val_loss: 0.3281
Epoch 26/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 1.0000 - loss: 6.7076e-09 - val
_accuracy: 0.9099 - val_loss: 0.3319
Epoch 27/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 6s 9ms/step - accuracy: 1.0000 - loss: 4.8174e-09 - val
_accuracy: 0.9093 - val_loss: 0.3353
Epoch 28/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 1.0000 - loss: 3.0524e-09 - val
_accuracy: 0.9099 - val_loss: 0.3381
Epoch 29/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 1.0000 - loss: 2.2958e-09 - val
_accuracy: 0.9099 - val_loss: 0.3407
Epoch 30/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 1.0000 - loss: 9.9090e-10 - val
_accuracy: 0.9093 - val_loss: 0.3434
Epoch 31/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 1.0000 - loss: 8.3106e-10 - val
_accuracy: 0.9099 - val_loss: 0.3455
Epoch 32/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 1.0000 - loss: 4.3117e-10 - val
_accuracy: 0.9099 - val_loss: 0.3477
Epoch 33/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 6s 9ms/step - accuracy: 1.0000 - loss: 1.5877e-10 - val
_accuracy: 0.9099 - val_loss: 0.3497
Epoch 34/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 1.0000 - loss: 1.1686e-10 - va
l_accuracy: 0.9105 - val_loss: 0.3514
Epoch 35/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 1.0000 - loss: 9.5189e-12 - val
_accuracy: 0.9099 - val_loss: 0.3530
Epoch 36/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val
_accuracy: 0.9105 - val_loss: 0.3544
Epoch 37/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 6s 9ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val
_accuracy: 0.9105 - val_loss: 0.3558
Epoch 38/50
```

```
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val
_accuracy: 0.9105 - val_loss: 0.3573
Epoch 39/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val
_accuracy: 0.9110 - val_loss: 0.3588
Epoch 40/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val
_accuracy: 0.9110 - val_loss: 0.3602
Epoch 41/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val
_accuracy: 0.9110 - val_loss: 0.3616
Epoch 42/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 6s 9ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val
_accuracy: 0.9110 - val_loss: 0.3628
Epoch 43/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val
_accuracy: 0.9110 - val_loss: 0.3640
Epoch 44/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val
_accuracy: 0.9121 - val_loss: 0.3650
Epoch 45/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val
_accuracy: 0.9127 - val_loss: 0.3660
Epoch 46/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val
_accuracy: 0.9127 - val_loss: 0.3669
Epoch 47/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 6s 9ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val
_accuracy: 0.9127 - val_loss: 0.3677
Epoch 48/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val
_accuracy: 0.9127 - val_loss: 0.3685
Epoch 49/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 8s 11ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - va
l_accuracy: 0.9127 - val_loss: 0.3692
Epoch 50/50
715/715 ━━━━━━━━━━━━━━━━━━━━ 8s 12ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - va
l_accuracy: 0.9127 - val_loss: 0.3698
```

In [82]:

```python
# Avaliar o modelo no conjunto de teste
nn_loss, nn_accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f'\nPrecisão da Rede Neural no Teste: {nn_accuracy}')
```

```
Precisão da Rede Neural no Teste: 0.917493462562561
```

In [83]:

```python
# Plotar a perda e a precisão durante o treinamento
plt.figure(figsize=(14, 6))
```

Out[83]:

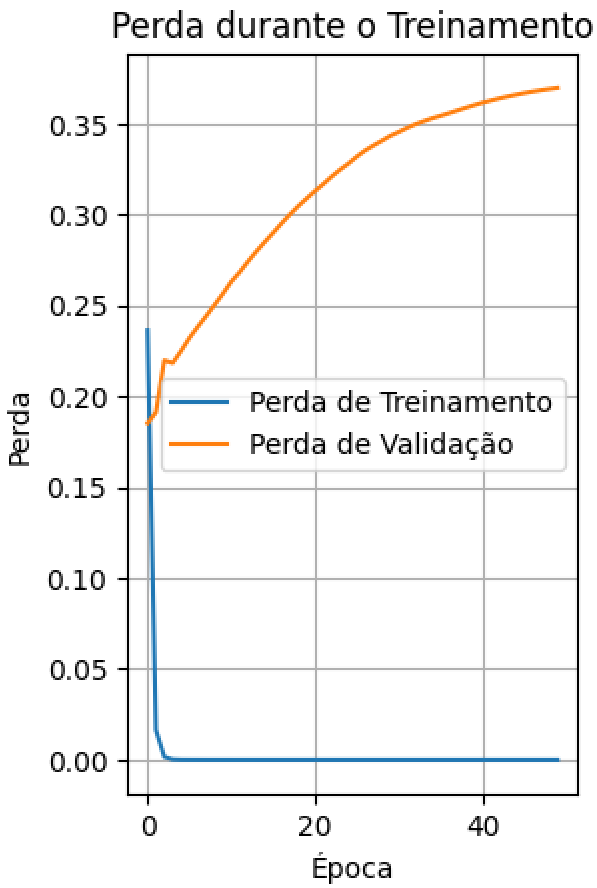
```
<Figure size 1400x600 with 0 Axes>
```

```
<Figure size 1400x600 with 0 Axes>
```

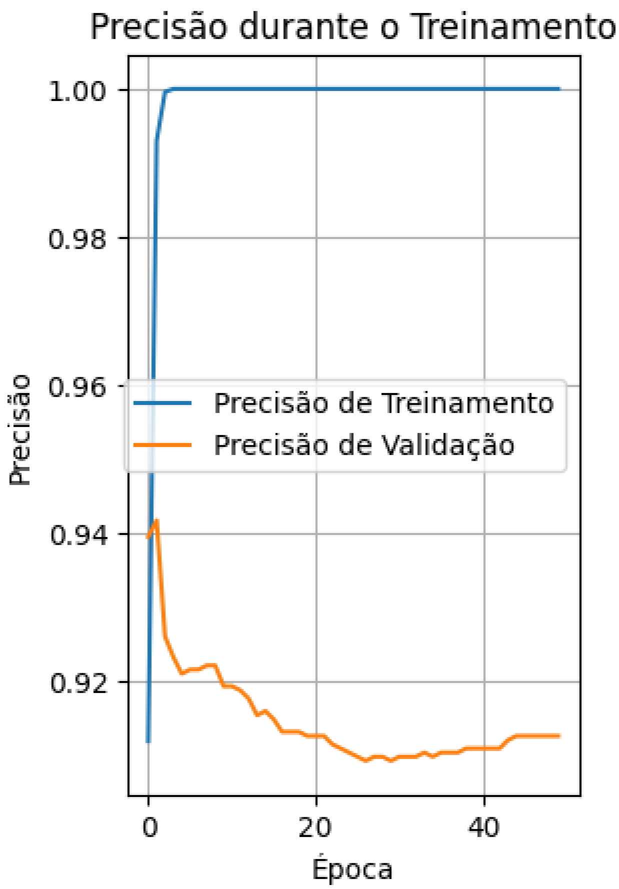
In [84]:

```python
# Plotar perda
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Perda de Treinamento')
plt.plot(history.history['val_loss'], label='Perda de Validação')
plt.title('Perda durante o Treinamento')
```

```
plt.xlabel('Época')
plt.ylabel('Perda')
plt.legend()
plt.grid(True)
```



In [85]:
```
# Plotar precisão
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Precisão de Treinamento')
plt.plot(history.history['val_accuracy'], label='Precisão de Validação')
plt.title('Precisão durante o Treinamento')
plt.xlabel('Época')
plt.ylabel('Precisão')
plt.legend()
plt.grid(True)

plt.show()
```


Precisão durante o Treinamento
1.00
0.98
0.96
0.94
0.92
Precisão
Precisão de Treinamento
Precisão de Validação
0
20
40
Época