In [ ]:

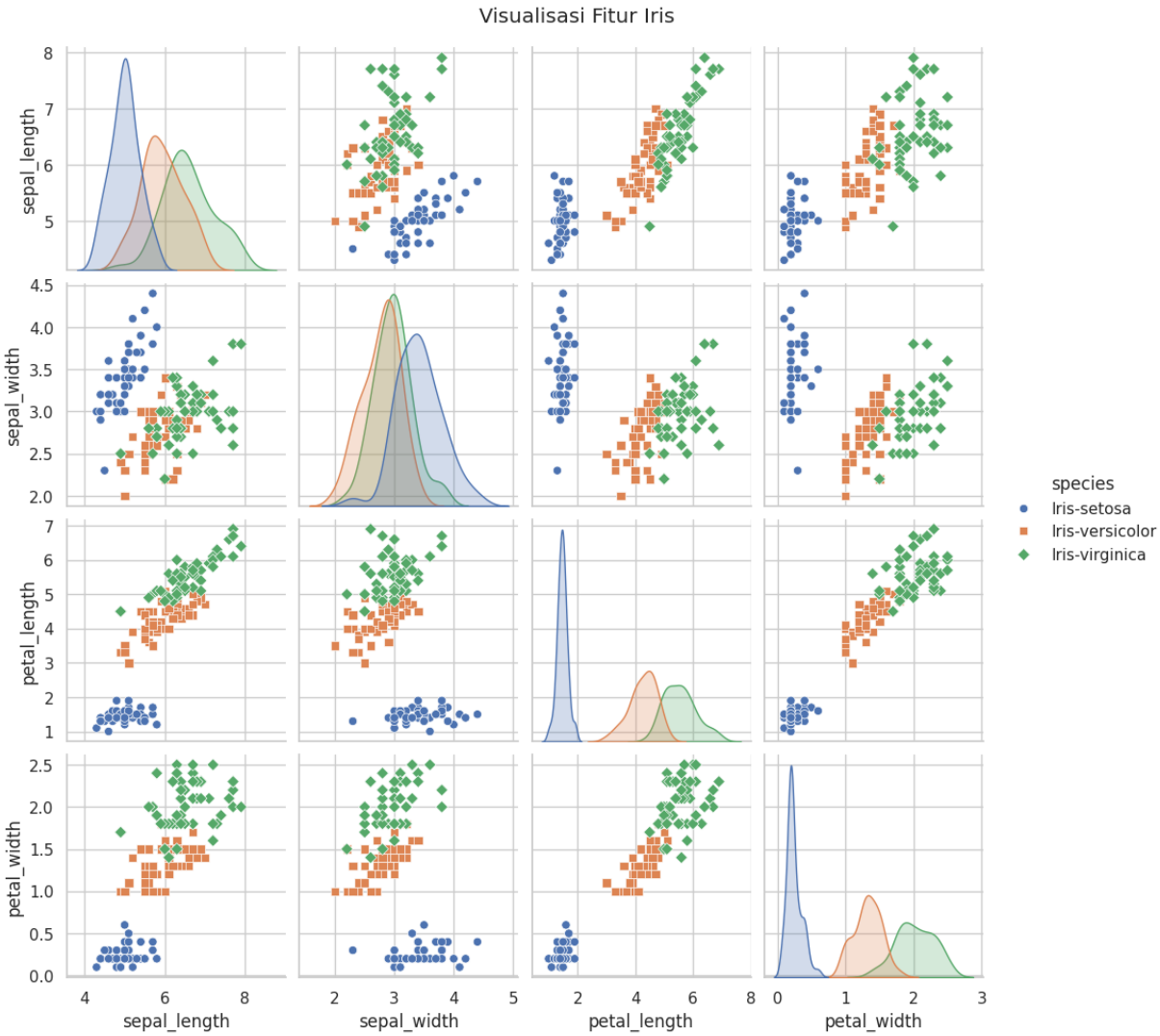
In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import classification_report, confusion_matrix
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.layers import Input


DATA_PATH = 'data/IRIS.csv'
OUTPUT_PATH = 'output/'
os.makedirs(OUTPUT_PATH, exist_ok=True)
sns.set(style="whitegrid")

print(f"TensorFlow Version: {tf.__version__}")
```

```
2025-11-30 16:13:17.443911: I external/local_xla/xla/tsl/cuda/cudart_stub.c
c:31] Could not find cuda drivers on your machine, GPU will not be used.
2025-11-30 16:13:29.356424: I tensorflow/core/platform/cpu_feature_guard.c
c:210] This TensorFlow binary is optimized to use available CPU instructions
in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild
TensorFlow with the appropriate compiler flags.
2025-11-30 16:13:54.796655: I external/local_xla/xla/tsl/cuda/cudart_stub.c
c:31] Could not find cuda drivers on your machine, GPU will not be used.
TensorFlow Version: 2.20.0
```

In [2]:
```
# Load Data
try:
    df = pd.read_csv(DATA_PATH)
    print("Dataset loaded successfully.")
except FileNotFoundError:
    sys.exit(f"Critical Error: {DATA_PATH} not found. Did you move the Kaggl

df.columns = df.columns.str.lower()

if 'id' in df.columns:
    df = df.drop(columns=['id'])

print("Columns found:", df.columns.tolist())
print(df.head())
```

```
sns.pairplot(df, hue="species", markers=["o", "s", "D"])
plt.suptitle("Visualisasi Fitur Iris", y=1.02)
plt.savefig(f"{OUTPUT_PATH}data_visualization.png")
plt.show()


target_col = 'species' if 'species' in df.columns else 'Species'
encoder = LabelEncoder()
y = encoder.fit_transform(df[target_col])
y_cat = to_categorical(y)

# Scaling Features
X = df.drop(target_col, axis=1).values
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split Data (80:20)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_cat, test_si
print(f"Training shape: {X_train.shape}, Testing shape: {X_test.shape}")
```

```
Dataset loaded successfully.
Columns found: ['sepal_length', 'sepal_width', 'petal_length', 'petal_width'
, 'species']
   sepal_length  sepal_width  petal_length  petal_width      species
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
```




```
Training shape: (120, 4), Testing shape: (30, 4)
```

In [3]:
```
model = Sequential([
    Input(shape=(4,)),
    Dense(16, activation='relu', name='Hidden_1'),
    Dense(16, activation='relu', name='Hidden_2'),
    Dense(3, activation='softmax', name='Output')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['a
model.summary()
```

```
2025-11-30 16:14:09.071213: E external/local_xla/xla/stream_executor/cuda/cu
da_platform.cc:51] failed call to cuInit: INTERNAL: CUDA error: Failed call
to cuInit: UNKNOWN ERROR (303)
```

Model: "sequential"

Layer (type)	Output Shape	Par
Hidden_1 (Dense)	(None, 16)	
Hidden_2 (Dense)	(None, 16)	
Output (Dense)	(None, 3)	

Total params: 403 (1.57 KB)

Trainable params: 403 (1.57 KB)

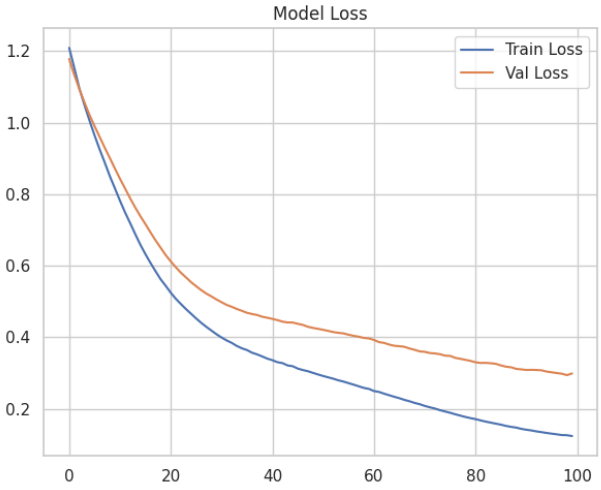
Non-trainable params: 0 (0.00 B)

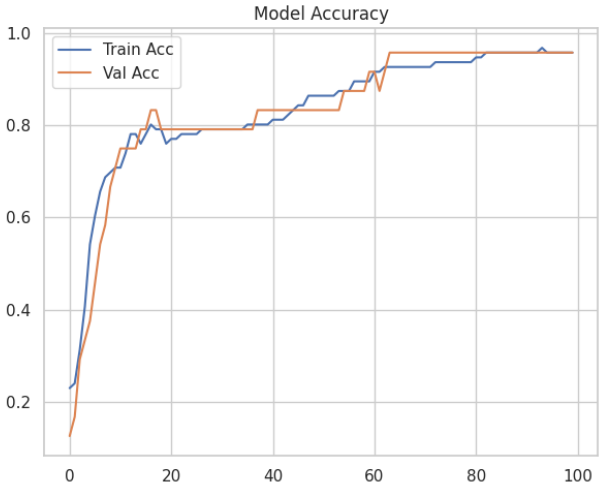
Bab 3: Hasil Eksperimen

In [4]:
```
# Training
history = model.fit(X_train, y_train, epochs=100, batch_size=16, validation_
print("Training selesai.")
```

```
Training selesai.
```

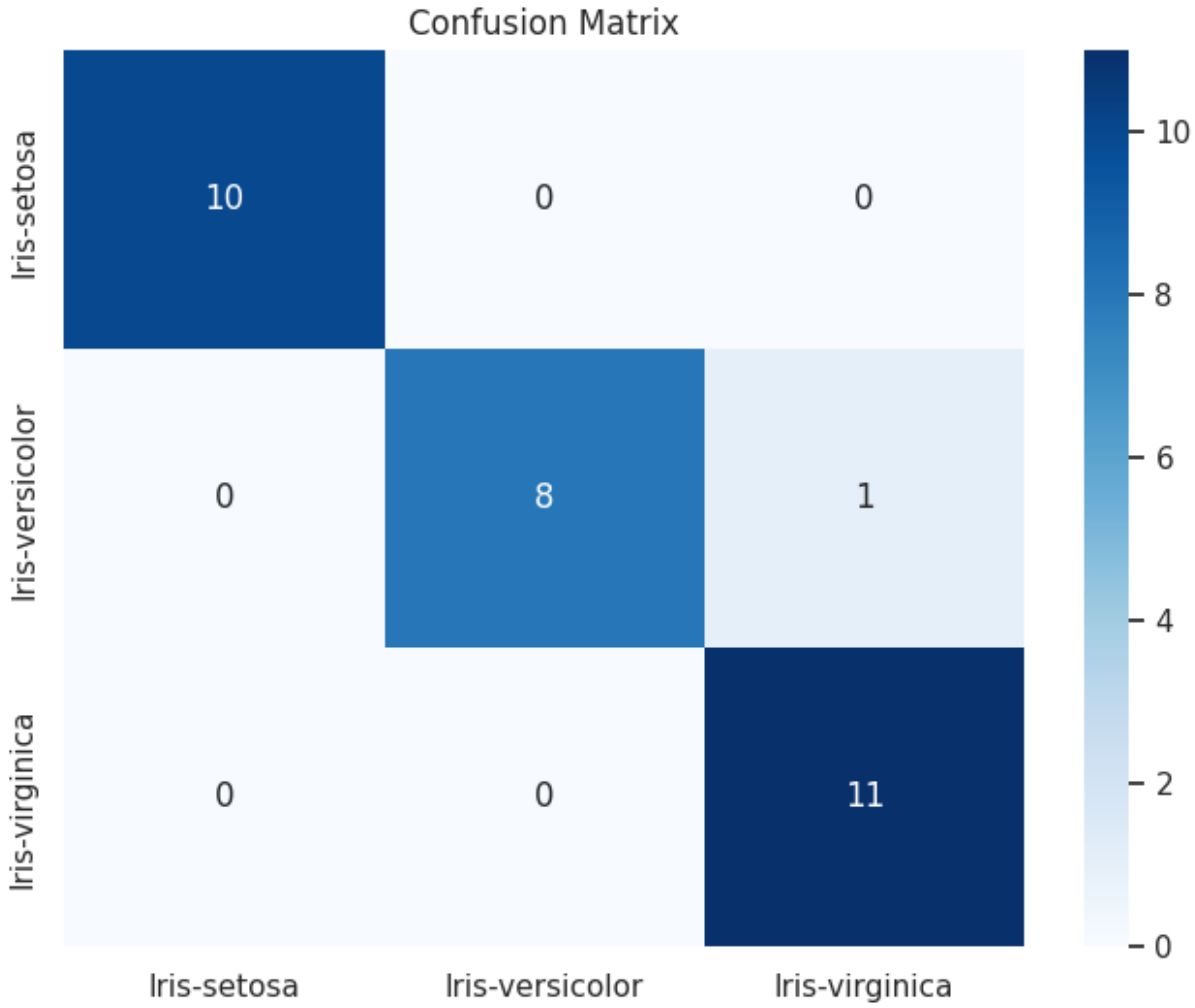
In [5]:
```
# Plot Loss & Accuracy
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title('Model Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Train Acc')
plt.plot(history.history['val_accuracy'], label='Val Acc')
plt.title('Model Accuracy')
plt.legend()

plt.tight_layout()
plt.savefig(f"{OUTPUT_PATH}training_history.png")
plt.show()
```





In [6]:
```
# Evaluasi & Confusion Matrix
loss, acc = model.evaluate(X_test, y_test, verbose=0)
print(f"\nFinal Test Accuracy: {acc*100:.2f}%")

y_pred = np.argmax(model.predict(X_test), axis=1)
y_true = np.argmax(y_test, axis=1)

cm = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=encoder.class
plt.title('Confusion Matrix')
plt.savefig(f"{OUTPUT_PATH}confusion_matrix.png")
plt.show()
```

```
Final Test Accuracy: 96.67%

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 69ms/step

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step
```


Confusion Matrix
Iris-setosa
Iris-versicolor
Iris-virginica
10
0
0
0
8
1
0
0
11
Iris-setosa
Iris-versicolor
Iris-virginica
10
8
6
4
2
0