Importado de librerias

In [1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

In [2]:
```
import pathlib
import tensorflow as tf
import matplotlib.pyplot as plt
import matplotlib.image as img
from sklearn.metrics import confusion_matrix
import numpy as np
import pandas as pd
import os
import PIL
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.applications import ResNet50, MobileNetV2
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam # - Works
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ModelCheckpoint
import random
from glob import glob
import seaborn as sns
from tensorflow.keras.losses import SparseCategoricalCrossentropy
import warnings
warnings.filterwarnings('ignore')
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
tf.compat.v1.logging.set_verbosity(tf.compat.v1.logging.ERROR)
```

Cargue de Imagenes

In [3]:
```
data_dir_train = pathlib.Path("/content/drive/My Drive/IA/dataset/Train/")
data_dir_test = pathlib.Path("/content/drive/My Drive/IA/dataset/Test/")
```

In [4]:
```
image_count_train = len(list(data_dir_train.glob('*/*.jpg')))
print(image_count_train)
```

```
image_count_test = len(list(data_dir_test.glob('*/*.jpg')))
print(image_count_test)
```

```
2239
118
```

Preparamos el DataSet

In [5]:
```
batch_size = 32
img_height = 224
img_width = 224
rnd_seed = 123
random.seed(rnd_seed)
EPOCHS = 50
```

In [6]:
```
dataset_training = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir_train,
  validation_split=0.15,
  subset="training",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size
)
```

```
Found 2239 files belonging to 9 classes.
Using 1904 files for training.
```

In [7]:
```
dataset_testing = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir_train,
  validation_split=0.15,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size
)
```

```
Found 2239 files belonging to 9 classes.
Using 335 files for validation.
```

In [8]:
```
test_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir_test,
  validation_split=0.99,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
```

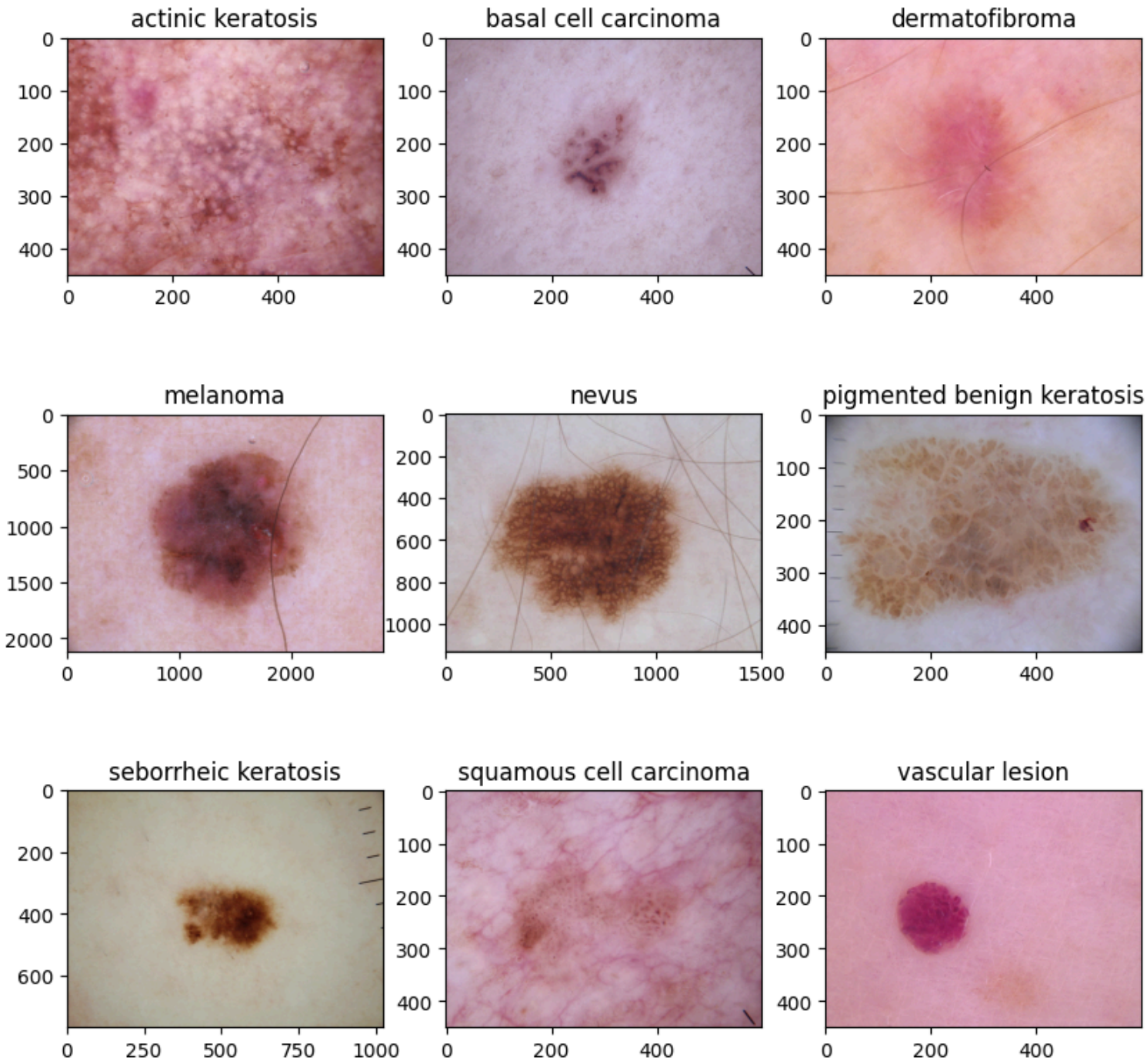
```
  batch_size=batch_size
)
```

```
Found 118 files belonging to 9 classes.
Using 116 files for validation.
```

In [9]:
```
class_names = dataset_training.class_names
print(class_names)
amount_classes = len(class_names)
```

```
['actinic keratosis', 'basal cell carcinoma', 'dermatofibroma', 'melanoma', 'nevus', 'pigmented benign keratosis', 'seborrheic kerat
osis', 'squamous cell carcinoma', 'vascular lesion']
```

In [10]:
```
num_classes = len(class_names)
plt.figure(figsize=(10,10))
for i in range(num_classes):
  plt.subplot(3,3,i+1)
  image = img.imread(str(list(data_dir_train.glob(class_names[i]+'/*.jpg'))[1]))
  plt.title(class_names[i])
  plt.imshow(image)
```


actinic keratosis
basal cell carcinoma
dermatofibroma
melanoma
nevus
pigmented benign keratosis
seborrheic keratosis
squamous cell carcinoma
vascular lesion

In [11]:
```
for image_batch, labels_batch in dataset_training.take(1):
    print(image_batch.shape)
    print(labels_batch.shape)
```

```
(32, 224, 224, 3)
(32,)
```

In [12]:
```
AUTOTUNE = tf.data.experimental.AUTOTUNE
dataset_training = dataset_training.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
dataset_testing = dataset_testing.cache().prefetch(buffer_size=AUTOTUNE)
```

Modelo 1 - Transfer Learning con ResNet50

In [13]:
```
num_classes = amount_classes

# Selección de modelo base (ResNet50 o MobileNetV2)
pretrained = ResNet50(weights='imagenet', include_top=False, input_shape=(img_height, img_width, 3))
# pretrained = MobileNetV2(weights='imagenet', include_top=False, input_shape=(img_height, img_width, 3))

pretrained.trainable = False

model = tf.keras.Sequential([
    tf.keras.layers.InputLayer(input_shape=(img_height, img_width, 3)),
    pretrained,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(num_classes, activation='softmax')
])
```

In [14]:
```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
resnet50 (Functional)	(None, 7, 7, 2048)	23,587,712
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
dense (Dense)	(None, 9)	18,441

Total params: 23,606,153 (90.05 MB)

Trainable params: 18,441 (72.04 KB)

Non-trainable params: 23,587,712 (89.98 MB)

In [15]:
```
optimizer = Adam(0.0001)

model.compile(
    optimizer=optimizer,
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=['accuracy']
)
```

In [16]:
```
lr_reduce = ReduceLROnPlateau(monitor='val_accuracy', factor=0.5, patience=2,mode='max', min_lr=0.00001,verbose=1)
early_stop = EarlyStopping(monitor="val_loss", patience=2, verbose=1)
model_chkpt = ModelCheckpoint('/content/drive/My Drive/IA/best_skin_cancer_model.keras',save_best_only=True, monitor='val_accuracy',

callback_list = [model_chkpt,lr_reduce]
```

In [17]:
```
historial = model.fit(
    dataset_training,
    epochs=EPOCHS,
    batch_size=batch_size,
    validation_data=dataset_testing,
    callbacks=callback_list
)
```

```
Epoch 1/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 148ms/step - accuracy: 0.1396 - loss: 2.4382
Epoch 1: val_accuracy improved from -inf to 0.21791, saving model to /content/drive/My Drive/IA/best_skin_cancer_model.keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 47s 321ms/step - accuracy: 0.1403 - loss: 2.4353 - val_accuracy: 0.2179 - val_loss: 2.1055 - learning_rat
e: 1.0000e-04
Epoch 2/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 83ms/step - accuracy: 0.2719 - loss: 2.0338
Epoch 2: val_accuracy improved from 0.21791 to 0.27164, saving model to /content/drive/My Drive/IA/best_skin_cancer_model.keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 7s 121ms/step - accuracy: 0.2724 - loss: 2.0324 - val_accuracy: 0.2716 - val_loss: 1.9251 - learning_rat
e: 1.0000e-04
Epoch 3/50
59/60 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step - accuracy: 0.3547 - loss: 1.7992
Epoch 3: val_accuracy improved from 0.27164 to 0.33731, saving model to /content/drive/My Drive/IA/best_skin_cancer_model.keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 7s 119ms/step - accuracy: 0.3549 - loss: 1.7989 - val_accuracy: 0.3373 - val_loss: 1.8028 - learning_rat
e: 1.0000e-04
Epoch 4/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.3988 - loss: 1.7069
Epoch 4: val_accuracy improved from 0.33731 to 0.39403, saving model to /content/drive/My Drive/IA/best_skin_cancer_model.keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 119ms/step - accuracy: 0.3991 - loss: 1.7064 - val_accuracy: 0.3940 - val_loss: 1.7159 - learning_rat
e: 1.0000e-04
Epoch 5/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.4431 - loss: 1.5945
Epoch 5: val_accuracy improved from 0.39403 to 0.40000, saving model to /content/drive/My Drive/IA/best_skin_cancer_model.keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 11s 123ms/step - accuracy: 0.4431 - loss: 1.5944 - val_accuracy: 0.4000 - val_loss: 1.6566 - learning_rat
e: 1.0000e-04
Epoch 6/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 85ms/step - accuracy: 0.4920 - loss: 1.4915
Epoch 6: val_accuracy improved from 0.40000 to 0.44179, saving model to /content/drive/My Drive/IA/best_skin_cancer_model.keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 123ms/step - accuracy: 0.4918 - loss: 1.4919 - val_accuracy: 0.4418 - val_loss: 1.6039 - learning_rat
e: 1.0000e-04
Epoch 7/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step - accuracy: 0.5117 - loss: 1.4410
Epoch 7: val_accuracy improved from 0.44179 to 0.45672, saving model to /content/drive/My Drive/IA/best_skin_cancer_model.keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 7s 124ms/step - accuracy: 0.5115 - loss: 1.4413 - val_accuracy: 0.4567 - val_loss: 1.5641 - learning_rat
e: 1.0000e-04
Epoch 8/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step - accuracy: 0.5244 - loss: 1.4195
Epoch 8: val_accuracy did not improve from 0.45672
60/60 ━━━━━━━━━━━━━━━━━━━━ 6s 104ms/step - accuracy: 0.5244 - loss: 1.4194 - val_accuracy: 0.4567 - val_loss: 1.5306 - learning_rat
e: 1.0000e-04
Epoch 9/50
59/60 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step - accuracy: 0.5524 - loss: 1.3644
Epoch 9: val_accuracy improved from 0.45672 to 0.46567, saving model to /content/drive/My Drive/IA/best_skin_cancer_model.keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 7s 119ms/step - accuracy: 0.5522 - loss: 1.3645 - val_accuracy: 0.4657 - val_loss: 1.4996 - learning_rat
e: 1.0000e-04
Epoch 10/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step - accuracy: 0.5352 - loss: 1.3603
```

```
Epoch 10: val_accuracy improved from 0.46567 to 0.47164, saving model to /content/drive/My Drive/IA/best_skin_cancer_model.keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 7s 119ms/step - accuracy: 0.5355 - loss: 1.3598 - val_accuracy: 0.4716 - val_loss: 1.4806 - learning_rat
e: 1.0000e-04
Epoch 11/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.5779 - loss: 1.2873
Epoch 11: val_accuracy improved from 0.47164 to 0.48060, saving model to /content/drive/My Drive/IA/best_skin_cancer_model.keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 8s 126ms/step - accuracy: 0.5778 - loss: 1.2875 - val_accuracy: 0.4806 - val_loss: 1.4544 - learning_rat
e: 1.0000e-04
Epoch 12/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step - accuracy: 0.5844 - loss: 1.2777
Epoch 12: val_accuracy improved from 0.48060 to 0.48358, saving model to /content/drive/My Drive/IA/best_skin_cancer_model.keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 7s 119ms/step - accuracy: 0.5843 - loss: 1.2775 - val_accuracy: 0.4836 - val_loss: 1.4380 - learning_rat
e: 1.0000e-04
Epoch 13/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step - accuracy: 0.5747 - loss: 1.2457
Epoch 13: val_accuracy improved from 0.48358 to 0.48955, saving model to /content/drive/My Drive/IA/best_skin_cancer_model.keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 7s 117ms/step - accuracy: 0.5749 - loss: 1.2456 - val_accuracy: 0.4896 - val_loss: 1.4234 - learning_rat
e: 1.0000e-04
Epoch 14/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.6185 - loss: 1.2041
Epoch 14: val_accuracy improved from 0.48955 to 0.49552, saving model to /content/drive/My Drive/IA/best_skin_cancer_model.keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 119ms/step - accuracy: 0.6183 - loss: 1.2043 - val_accuracy: 0.4955 - val_loss: 1.4063 - learning_rat
e: 1.0000e-04
Epoch 15/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.6153 - loss: 1.1829
Epoch 15: val_accuracy improved from 0.49552 to 0.50448, saving model to /content/drive/My Drive/IA/best_skin_cancer_model.keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 7s 118ms/step - accuracy: 0.6152 - loss: 1.1831 - val_accuracy: 0.5045 - val_loss: 1.3890 - learning_rat
e: 1.0000e-04
Epoch 16/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.6019 - loss: 1.1767
Epoch 16: val_accuracy did not improve from 0.50448
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 95ms/step - accuracy: 0.6021 - loss: 1.1766 - val_accuracy: 0.4985 - val_loss: 1.3855 - learning_rate:
1.0000e-04
Epoch 17/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step - accuracy: 0.6199 - loss: 1.1277
Epoch 17: val_accuracy improved from 0.50448 to 0.51642, saving model to /content/drive/My Drive/IA/best_skin_cancer_model.keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 12s 118ms/step - accuracy: 0.6200 - loss: 1.1280 - val_accuracy: 0.5164 - val_loss: 1.3704 - learning_rat
e: 1.0000e-04
Epoch 18/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.6261 - loss: 1.1513
Epoch 18: val_accuracy did not improve from 0.51642
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 95ms/step - accuracy: 0.6262 - loss: 1.1510 - val_accuracy: 0.5045 - val_loss: 1.3615 - learning_rate:
1.0000e-04
Epoch 19/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step - accuracy: 0.6459 - loss: 1.1067
Epoch 19: val_accuracy did not improve from 0.51642
```

```
Epoch 19: ReduceLROnPlateau reducing learning rate to 4.999999873689376e-05.
60/60 ━━━━━━━━━━━━━━━━━━━━ 6s 97ms/step - accuracy: 0.6458 - loss: 1.1068 - val_accuracy: 0.5104 - val_loss: 1.3508 - learning_rate:
1.0000e-04
Epoch 20/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.6486 - loss: 1.0876
Epoch 20: val_accuracy did not improve from 0.51642
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 95ms/step - accuracy: 0.6485 - loss: 1.0877 - val_accuracy: 0.5164 - val_loss: 1.3463 - learning_rat
e: 5.0000e-05
Epoch 21/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.6427 - loss: 1.1128
Epoch 21: val_accuracy did not improve from 0.51642

Epoch 21: ReduceLROnPlateau reducing learning rate to 2.499999936844688e-05.
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 95ms/step - accuracy: 0.6428 - loss: 1.1124 - val_accuracy: 0.5134 - val_loss: 1.3426 - learning_rat
e: 5.0000e-05
Epoch 22/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.6347 - loss: 1.0816
Epoch 22: val_accuracy did not improve from 0.51642
60/60 ━━━━━━━━━━━━━━━━━━━━ 6s 95ms/step - accuracy: 0.6349 - loss: 1.0815 - val_accuracy: 0.5164 - val_loss: 1.3406 - learning_rate:
2.5000e-05
Epoch 23/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.6581 - loss: 1.0599
Epoch 23: val_accuracy did not improve from 0.51642

Epoch 23: ReduceLROnPlateau reducing learning rate to 1.249999968422344e-05.
60/60 ━━━━━━━━━━━━━━━━━━━━ 6s 95ms/step - accuracy: 0.6580 - loss: 1.0601 - val_accuracy: 0.5134 - val_loss: 1.3391 - learning_rate:
2.5000e-05
Epoch 24/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step - accuracy: 0.6522 - loss: 1.0677
Epoch 24: val_accuracy did not improve from 0.51642
60/60 ━━━━━━━━━━━━━━━━━━━━ 6s 95ms/step - accuracy: 0.6522 - loss: 1.0677 - val_accuracy: 0.5164 - val_loss: 1.3378 - learning_rate:
1.2500e-05
Epoch 25/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step - accuracy: 0.6729 - loss: 1.0322
Epoch 25: val_accuracy improved from 0.51642 to 0.51940, saving model to /content/drive/My Drive/IA/best_skin_cancer_model.keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 7s 119ms/step - accuracy: 0.6726 - loss: 1.0328 - val_accuracy: 0.5194 - val_loss: 1.3370 - learning_rat
e: 1.2500e-05
Epoch 26/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step - accuracy: 0.6636 - loss: 1.0501
Epoch 26: val_accuracy did not improve from 0.51940
60/60 ━━━━━━━━━━━━━━━━━━━━ 6s 94ms/step - accuracy: 0.6634 - loss: 1.0504 - val_accuracy: 0.5164 - val_loss: 1.3357 - learning_rate:
1.2500e-05
Epoch 27/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.6596 - loss: 1.0559
Epoch 27: val_accuracy improved from 0.51940 to 0.52239, saving model to /content/drive/My Drive/IA/best_skin_cancer_model.keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 12s 118ms/step - accuracy: 0.6595 - loss: 1.0561 - val_accuracy: 0.5224 - val_loss: 1.3352 - learning_rat
e: 1.2500e-05
```

```
Epoch 28/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step - accuracy: 0.6484 - loss: 1.0656
Epoch 28: val_accuracy did not improve from 0.52239
60/60 ━━━━━━━━━━━━━━━━━━━━ 6s 94ms/step - accuracy: 0.6485 - loss: 1.0655 - val_accuracy: 0.5194 - val_loss: 1.3339 - learning_rate:
1.2500e-05
Epoch 29/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.6336 - loss: 1.0819
Epoch 29: val_accuracy did not improve from 0.52239

Epoch 29: ReduceLROnPlateau reducing learning rate to 1e-05.
60/60 ━━━━━━━━━━━━━━━━━━━━ 6s 102ms/step - accuracy: 0.6339 - loss: 1.0815 - val_accuracy: 0.5224 - val_loss: 1.3332 - learning_rat
e: 1.2500e-05
Epoch 30/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.6592 - loss: 1.0413
Epoch 30: val_accuracy did not improve from 0.52239
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 95ms/step - accuracy: 0.6591 - loss: 1.0416 - val_accuracy: 0.5164 - val_loss: 1.3326 - learning_rat
e: 1.0000e-05
Epoch 31/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step - accuracy: 0.6597 - loss: 1.0437
Epoch 31: val_accuracy did not improve from 0.52239
60/60 ━━━━━━━━━━━━━━━━━━━━ 6s 94ms/step - accuracy: 0.6596 - loss: 1.0439 - val_accuracy: 0.5224 - val_loss: 1.3315 - learning_rate:
1.0000e-05
Epoch 32/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.6614 - loss: 1.0585
Epoch 32: val_accuracy did not improve from 0.52239
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 95ms/step - accuracy: 0.6613 - loss: 1.0584 - val_accuracy: 0.5194 - val_loss: 1.3315 - learning_rat
e: 1.0000e-05
Epoch 33/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.6445 - loss: 1.0739
Epoch 33: val_accuracy did not improve from 0.52239
60/60 ━━━━━━━━━━━━━━━━━━━━ 6s 96ms/step - accuracy: 0.6447 - loss: 1.0735 - val_accuracy: 0.5224 - val_loss: 1.3300 - learning_rate:
1.0000e-05
Epoch 34/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.6595 - loss: 1.0506
Epoch 34: val_accuracy improved from 0.52239 to 0.52537, saving model to /content/drive/My Drive/IA/best_skin_cancer_model.keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 12s 121ms/step - accuracy: 0.6595 - loss: 1.0506 - val_accuracy: 0.5254 - val_loss: 1.3293 - learning_rat
e: 1.0000e-05
Epoch 35/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step - accuracy: 0.6556 - loss: 1.0543
Epoch 35: val_accuracy did not improve from 0.52537
60/60 ━━━━━━━━━━━━━━━━━━━━ 6s 96ms/step - accuracy: 0.6556 - loss: 1.0543 - val_accuracy: 0.5254 - val_loss: 1.3288 - learning_rate:
1.0000e-05
Epoch 36/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step - accuracy: 0.6325 - loss: 1.0663
Epoch 36: val_accuracy did not improve from 0.52537
60/60 ━━━━━━━━━━━━━━━━━━━━ 11s 102ms/step - accuracy: 0.6329 - loss: 1.0660 - val_accuracy: 0.5224 - val_loss: 1.3282 - learning_rat
e: 1.0000e-05
```

```
Epoch 37/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step - accuracy: 0.6567 - loss: 1.0552
Epoch 37: val_accuracy did not improve from 0.52537
60/60 ━━━━━━━━━━━━━━━━━━━━ 6s 94ms/step - accuracy: 0.6568 - loss: 1.0551 - val_accuracy: 0.5254 - val_loss: 1.3274 - learning_rate:
1.0000e-05
Epoch 38/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.6725 - loss: 1.0203
Epoch 38: val_accuracy did not improve from 0.52537
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 95ms/step - accuracy: 0.6722 - loss: 1.0208 - val_accuracy: 0.5254 - val_loss: 1.3269 - learning_rat
e: 1.0000e-05
Epoch 39/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.6677 - loss: 1.0188
Epoch 39: val_accuracy did not improve from 0.52537
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 95ms/step - accuracy: 0.6675 - loss: 1.0192 - val_accuracy: 0.5254 - val_loss: 1.3264 - learning_rat
e: 1.0000e-05
Epoch 40/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.6587 - loss: 1.0477
Epoch 40: val_accuracy did not improve from 0.52537
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 95ms/step - accuracy: 0.6587 - loss: 1.0476 - val_accuracy: 0.5254 - val_loss: 1.3252 - learning_rat
e: 1.0000e-05
Epoch 41/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 88ms/step - accuracy: 0.6641 - loss: 1.0437
Epoch 41: val_accuracy did not improve from 0.52537
60/60 ━━━━━━━━━━━━━━━━━━━━ 11s 110ms/step - accuracy: 0.6640 - loss: 1.0437 - val_accuracy: 0.5254 - val_loss: 1.3249 - learning_rat
e: 1.0000e-05
Epoch 42/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.6552 - loss: 1.0449
Epoch 42: val_accuracy did not improve from 0.52537
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 95ms/step - accuracy: 0.6553 - loss: 1.0448 - val_accuracy: 0.5224 - val_loss: 1.3242 - learning_rate:
1.0000e-05
Epoch 43/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.6608 - loss: 1.0491
Epoch 43: val_accuracy did not improve from 0.52537
60/60 ━━━━━━━━━━━━━━━━━━━━ 6s 95ms/step - accuracy: 0.6608 - loss: 1.0489 - val_accuracy: 0.5254 - val_loss: 1.3233 - learning_rate:
1.0000e-05
Epoch 44/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step - accuracy: 0.6685 - loss: 1.0314
Epoch 44: val_accuracy did not improve from 0.52537
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 95ms/step - accuracy: 0.6684 - loss: 1.0315 - val_accuracy: 0.5254 - val_loss: 1.3228 - learning_rat
e: 1.0000e-05
Epoch 45/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.6573 - loss: 1.0466
Epoch 45: val_accuracy did not improve from 0.52537
60/60 ━━━━━━━━━━━━━━━━━━━━ 6s 103ms/step - accuracy: 0.6574 - loss: 1.0464 - val_accuracy: 0.5254 - val_loss: 1.3221 - learning_rat
e: 1.0000e-05
Epoch 46/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step - accuracy: 0.6644 - loss: 1.0155
```

```
Epoch 46: val_accuracy did not improve from 0.52537
60/60 ━━━━━━━━━━━━━━━━━━━━ 6s 94ms/step - accuracy: 0.6644 - loss: 1.0158 - val_accuracy: 0.5254 - val_loss: 1.3216 - learning_rate:
1.0000e-05
Epoch 47/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step - accuracy: 0.6572 - loss: 1.0278
Epoch 47: val_accuracy did not improve from 0.52537
60/60 ━━━━━━━━━━━━━━━━━━━━ 6s 95ms/step - accuracy: 0.6572 - loss: 1.0279 - val_accuracy: 0.5224 - val_loss: 1.3204 - learning_rate:
1.0000e-05
Epoch 48/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step - accuracy: 0.6695 - loss: 1.0135
Epoch 48: val_accuracy did not improve from 0.52537
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 94ms/step - accuracy: 0.6694 - loss: 1.0138 - val_accuracy: 0.5254 - val_loss: 1.3202 - learning_rat
e: 1.0000e-05
Epoch 49/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.6364 - loss: 1.0525
Epoch 49: val_accuracy did not improve from 0.52537
60/60 ━━━━━━━━━━━━━━━━━━━━ 6s 95ms/step - accuracy: 0.6368 - loss: 1.0521 - val_accuracy: 0.5254 - val_loss: 1.3193 - learning_rate:
1.0000e-05
Epoch 50/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 79ms/step - accuracy: 0.6658 - loss: 1.0430
Epoch 50: val_accuracy did not improve from 0.52537
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 94ms/step - accuracy: 0.6657 - loss: 1.0428 - val_accuracy: 0.5254 - val_loss: 1.3192 - learning_rat
e: 1.0000e-05
```

Graficas del Primer Modelo

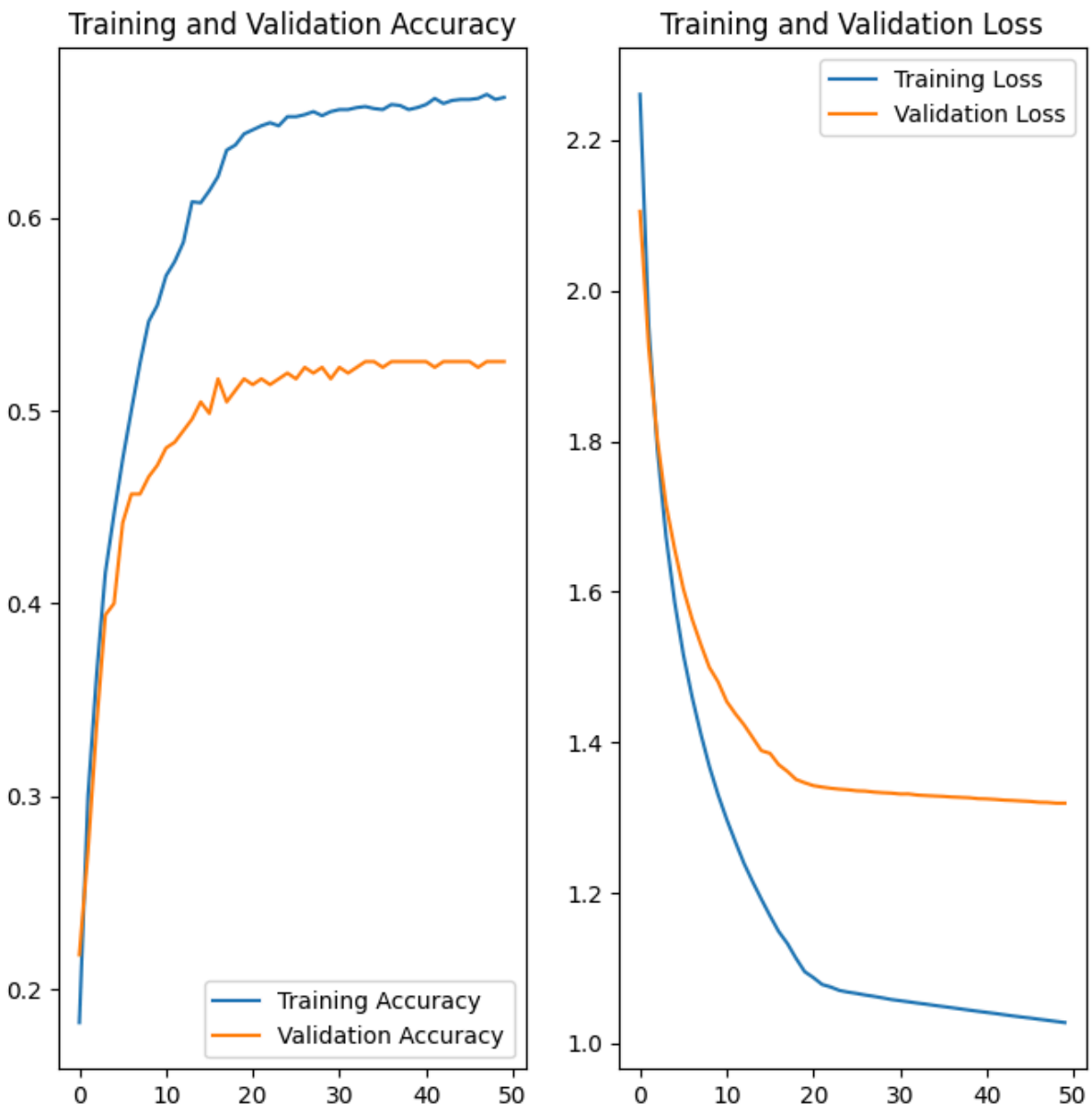
In [18]:

```python
def mostrar_matriz_confusion(y_true, y_pred, labels=None, normalize='true'):
    # Generar la matriz de confusión
    cm = confusion_matrix(y_true, y_pred, labels=labels, normalize=normalize)

    # Crear el gráfico
    plt.figure(figsize=(8, 6))
    plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
    plt.title("Matriz de Confusión")
    plt.colorbar()

    tick_marks = np.arange(len(labels))
    plt.xticks(tick_marks, labels, rotation=45)
    plt.yticks(tick_marks, labels)

    # Agregar etiquetas
    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
```

```
            plt.text(j, i, format(cm[i, j], fmt),
                     ha="center", va="center",
                     color="white" if cm[i, j] > thresh else "black")

    plt.ylabel('Etiqueta Real')
    plt.xlabel('Etiqueta Predicha')
    plt.tight_layout()
    plt.show()
```

In [19]:

```
acc = historial.history['accuracy']
val_acc = historial.history['val_accuracy']

loss = historial.history['loss']
val_loss = historial.history['val_loss']

epochs_range = range(EPOCHS)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



In [20]:
```
# Evaluar el modelo y obtener predicciones en el conjunto de prueba
y_true = np.concatenate([y for x, y in dataset_testing], axis=0)  # Etiquetas reales del conjunto de prueba
y_pred = model.predict(dataset_testing)  # Predicciones del modelo
y_pred = np.argmax(y_pred, axis=1)  # Convierte las probabilidades en etiquetas si es softmax
```

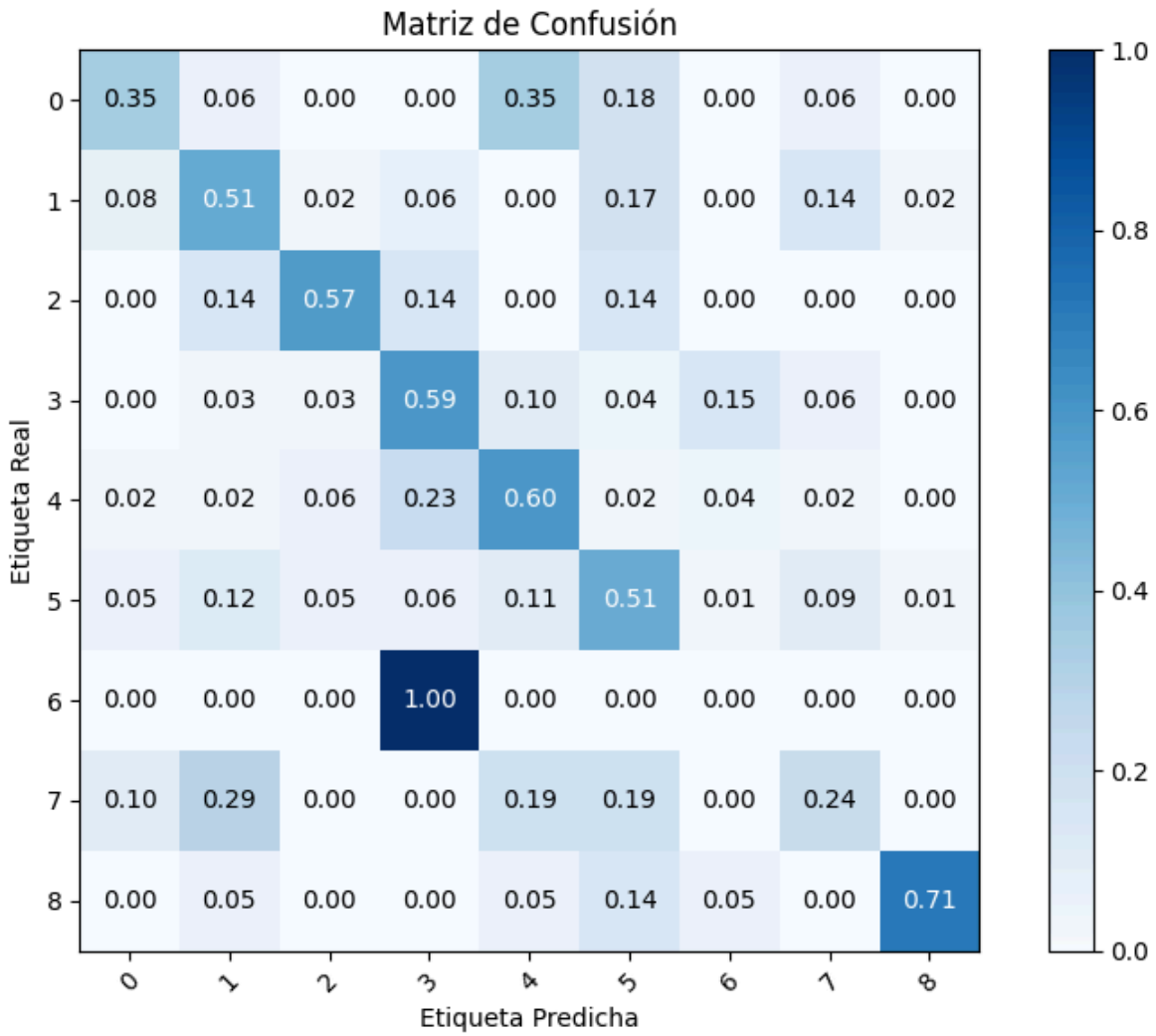
```
label_to_index = {label: idx for idx, label in enumerate(class_names)}
labels = list(label_to_index.values())  # Esto toma solo los valores numéricos del diccionario
labels
```

```
11/11 ━━━━━━━━━━━━━━━━━━━━ 10s 702ms/step
```

Out[20]: [0, 1, 2, 3, 4, 5, 6, 7, 8]

In [21]:

```
# Mostrar la matriz de confusión
mostrar_matriz_confusion(y_pred, y_true, labels=labels)  # Cambia los labels según tus clases
```



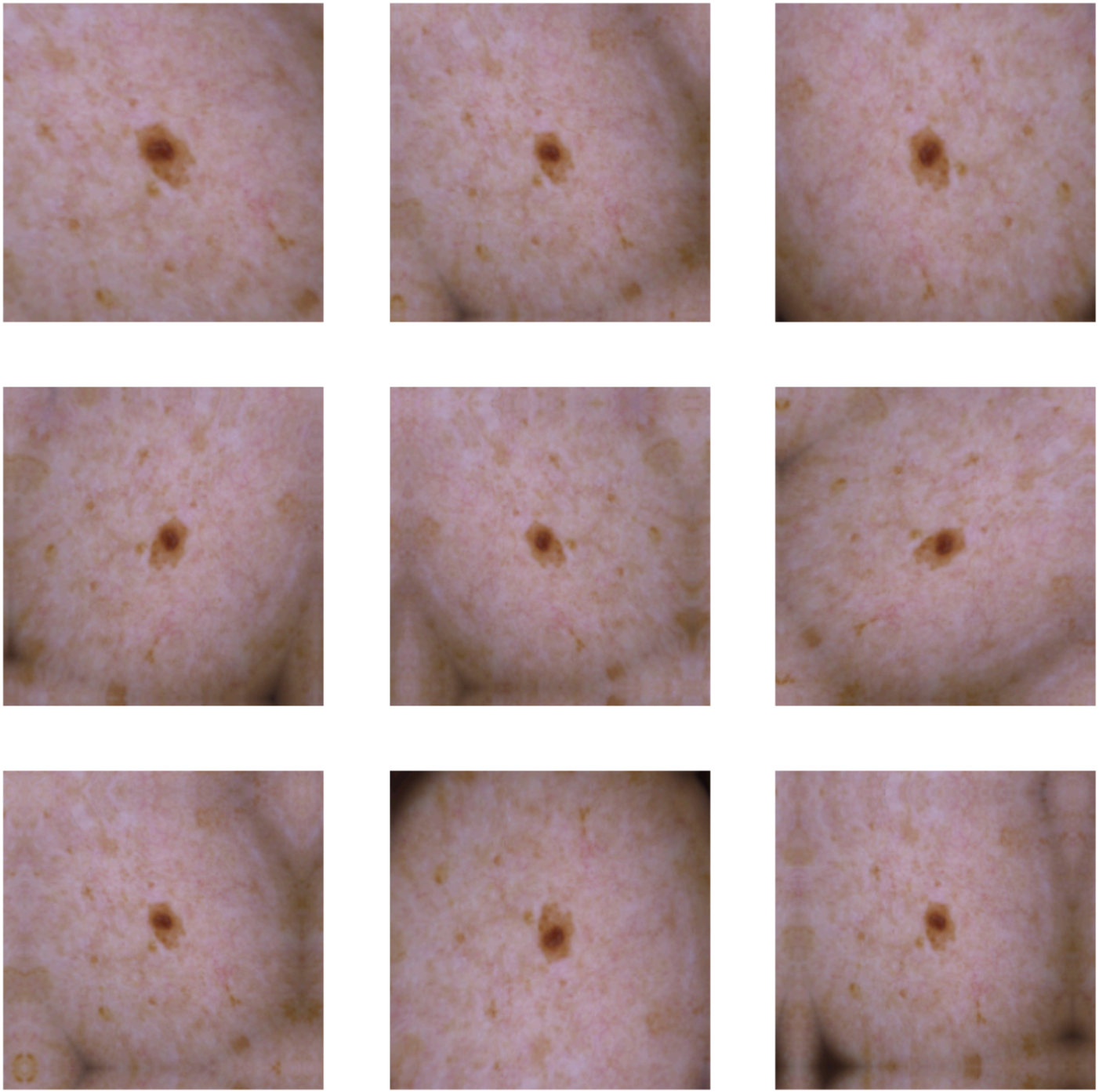
Análisis :

- El modelo muestra una buena precisión en el conjunto de entrenamiento, alcanzando hasta un 70%, mientras que en datos nuevos (validación) la precisión baja ligeramente a un 60%, lo cual sugiere que podría estar "memoriza" los datos de entrenamiento en lugar de generalizar bien a casos nuevos.
- En la matriz de confusión, se observa que el modelo acierta en algunas lesiones, como "nevus" y "vascular lesion", pero comete errores en otras, confundiendo tipos de lesiones que se ven similares, como "actinic keratosis" con "nevus".

Modelo con Regularización y Aumento de Datos (Random Flip, Rotation y Zoom)

In [22]:
```
data_augmentation = Sequential(
    [
        layers.RandomFlip("horizontal_and_vertical", input_shape=(img_height, img_width, 3)),
        layers.RandomRotation(0.2),
        layers.RandomZoom(0.2),
    ]
)
```

In [23]:
```
plt.figure(figsize=(10, 10))
for images, _ in dataset_training.take(1):
  for i in range(9):
    augmented_images = data_augmentation(images)
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(augmented_images[0].numpy().astype("uint8"))
    plt.axis("off")
```



In [29]:
```
num_classes = 9
model = tf.keras.Sequential([
    tf.keras.layers.InputLayer(input_shape=(img_height, img_width, 3)),
    data_augmentation,
    pretrained,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(num_classes, activation='softmax')
])
```

In [30]:
```
model.summary()
```

Model: "sequential_3"

Layer (type)	Output Shape	Param #
sequential_1 (Sequential)	(None, 224, 224, 3)	0
resnet50 (Functional)	(None, 7, 7, 2048)	23,587,712
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 2048)	0
dense_2 (Dense)	(None, 9)	18,441

Total params: 23,606,153 (90.05 MB)

Trainable params: 18,441 (72.04 KB)

Non-trainable params: 23,587,712 (89.98 MB)

In [31]:
```
# Definir el optimizador con el argumento correcto
opt = Adam(learning_rate=0.001)

# Compilar el modelo
model.compile(
    optimizer=opt,
    loss=SparseCategoricalCrossentropy(from_logits=True),
    metrics=['accuracy']
)
```

In [32]:
```
lr_reduce = ReduceLROnPlateau(monitor='val_accuracy', factor=0.5, patience=2,mode='max', min_lr=0.00001,verbose=1)
early_stop = EarlyStopping(monitor="val_loss", patience=2, verbose=1)
model_chkpt = ModelCheckpoint('/content/drive/My Drive/IA/best_skin_cancer_model_augmented.keras',save_best_only=True, monitor='val_

callback_list = [model_chkpt,lr_reduce]
```

In [33]:
```
history = model.fit(
    dataset_training,
    epochs=EPOCHS,
    batch_size=batch_size,
    validation_data=dataset_testing,
    callbacks=callback_list
)
```

```
Epoch 1/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 142ms/step - accuracy: 0.3120 - loss: 1.9354
Epoch 1: val_accuracy improved from -inf to 0.45672, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_augmented.ker
as
60/60 ━━━━━━━━━━━━━━━━━━━━ 24s 238ms/step - accuracy: 0.3134 - loss: 1.9319 - val_accuracy: 0.4567 - val_loss: 1.5517 - learning_rat
e: 0.0010
Epoch 2/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step - accuracy: 0.5026 - loss: 1.4035
Epoch 2: val_accuracy improved from 0.45672 to 0.50746, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_augmented.
keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 11s 182ms/step - accuracy: 0.5028 - loss: 1.4033 - val_accuracy: 0.5075 - val_loss: 1.4617 - learning_rat
e: 0.0010
Epoch 3/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 129ms/step - accuracy: 0.5686 - loss: 1.2674
Epoch 3: val_accuracy improved from 0.50746 to 0.53433, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_augmented.
keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 20s 178ms/step - accuracy: 0.5684 - loss: 1.2675 - val_accuracy: 0.5343 - val_loss: 1.4040 - learning_rat
e: 0.0010
Epoch 4/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 129ms/step - accuracy: 0.6175 - loss: 1.1647
Epoch 4: val_accuracy did not improve from 0.53433
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 151ms/step - accuracy: 0.6175 - loss: 1.1646 - val_accuracy: 0.5164 - val_loss: 1.4212 - learning_rat
e: 0.0010
Epoch 5/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 129ms/step - accuracy: 0.6210 - loss: 1.0951
Epoch 5: val_accuracy improved from 0.53433 to 0.54627, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_augmented.
keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 12s 175ms/step - accuracy: 0.6209 - loss: 1.0953 - val_accuracy: 0.5463 - val_loss: 1.3679 - learning_rat
e: 0.0010
Epoch 6/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 127ms/step - accuracy: 0.6377 - loss: 1.0748
Epoch 6: val_accuracy did not improve from 0.54627
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 150ms/step - accuracy: 0.6376 - loss: 1.0750 - val_accuracy: 0.5313 - val_loss: 1.4235 - learning_rat
e: 0.0010
Epoch 7/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 127ms/step - accuracy: 0.6280 - loss: 1.0500
Epoch 7: val_accuracy improved from 0.54627 to 0.57910, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_augmented.
keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 14s 206ms/step - accuracy: 0.6282 - loss: 1.0500 - val_accuracy: 0.5791 - val_loss: 1.4158 - learning_rat
e: 0.0010
Epoch 8/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 128ms/step - accuracy: 0.6617 - loss: 0.9908
Epoch 8: val_accuracy did not improve from 0.57910
60/60 ━━━━━━━━━━━━━━━━━━━━ 17s 150ms/step - accuracy: 0.6615 - loss: 0.9912 - val_accuracy: 0.5701 - val_loss: 1.3685 - learning_rat
e: 0.0010
Epoch 9/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 127ms/step - accuracy: 0.6444 - loss: 1.0077
```

```
Epoch 9: val_accuracy did not improve from 0.57910

Epoch 9: ReduceLROnPlateau reducing learning rate to 0.00050000000237487257.
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 150ms/step - accuracy: 0.6446 - loss: 1.0074 - val_accuracy: 0.5701 - val_loss: 1.3914 - learning_rat
e: 0.0010
Epoch 10/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step - accuracy: 0.6641 - loss: 0.9594
Epoch 10: val_accuracy did not improve from 0.57910
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 152ms/step - accuracy: 0.6641 - loss: 0.9594 - val_accuracy: 0.5582 - val_loss: 1.3453 - learning_rat
e: 5.0000e-04
Epoch 11/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step - accuracy: 0.6883 - loss: 0.9474
Epoch 11: val_accuracy did not improve from 0.57910

Epoch 11: ReduceLROnPlateau reducing learning rate to 0.00025000000118743628.
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 153ms/step - accuracy: 0.6881 - loss: 0.9472 - val_accuracy: 0.5612 - val_loss: 1.3167 - learning_rat
e: 5.0000e-04
Epoch 12/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step - accuracy: 0.6859 - loss: 0.9230
Epoch 12: val_accuracy did not improve from 0.57910
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 152ms/step - accuracy: 0.6861 - loss: 0.9228 - val_accuracy: 0.5731 - val_loss: 1.3267 - learning_rat
e: 2.5000e-04
Epoch 13/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 128ms/step - accuracy: 0.6882 - loss: 0.9008
Epoch 13: val_accuracy did not improve from 0.57910

Epoch 13: ReduceLROnPlateau reducing learning rate to 0.00012500000059371814.
60/60 ━━━━━━━━━━━━━━━━━━━━ 11s 171ms/step - accuracy: 0.6881 - loss: 0.9012 - val_accuracy: 0.5731 - val_loss: 1.3184 - learning_rat
e: 2.5000e-04
Epoch 14/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 126ms/step - accuracy: 0.6945 - loss: 0.9178
Epoch 14: val_accuracy improved from 0.57910 to 0.58209, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_augmente
d.keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 11s 179ms/step - accuracy: 0.6945 - loss: 0.9175 - val_accuracy: 0.5821 - val_loss: 1.3070 - learning_rat
e: 1.2500e-04
Epoch 15/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 127ms/step - accuracy: 0.6833 - loss: 0.9349
Epoch 15: val_accuracy did not improve from 0.58209
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 170ms/step - accuracy: 0.6834 - loss: 0.9345 - val_accuracy: 0.5821 - val_loss: 1.3023 - learning_rat
e: 1.2500e-04
Epoch 16/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 128ms/step - accuracy: 0.6923 - loss: 0.9121
Epoch 16: val_accuracy improved from 0.58209 to 0.59104, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_augmente
d.keras
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 173ms/step - accuracy: 0.6923 - loss: 0.9118 - val_accuracy: 0.5910 - val_loss: 1.2938 - learning_rat
e: 1.2500e-04
Epoch 17/50
```

```
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 128ms/step - accuracy: 0.7001 - loss: 0.8845
Epoch 17: val_accuracy did not improve from 0.59104
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 150ms/step - accuracy: 0.7001 - loss: 0.8846 - val_accuracy: 0.5851 - val_loss: 1.2977 - learning_rat
e: 1.2500e-04
Epoch 18/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 129ms/step - accuracy: 0.6953 - loss: 0.8950
Epoch 18: val_accuracy did not improve from 0.59104

Epoch 18: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 151ms/step - accuracy: 0.6954 - loss: 0.8948 - val_accuracy: 0.5821 - val_loss: 1.2968 - learning_rat
e: 1.2500e-04
Epoch 19/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step - accuracy: 0.6864 - loss: 0.9067
Epoch 19: val_accuracy did not improve from 0.59104
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 152ms/step - accuracy: 0.6865 - loss: 0.9065 - val_accuracy: 0.5881 - val_loss: 1.2891 - learning_rat
e: 6.2500e-05
Epoch 20/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 129ms/step - accuracy: 0.7011 - loss: 0.9028
Epoch 20: val_accuracy did not improve from 0.59104

Epoch 20: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 152ms/step - accuracy: 0.7011 - loss: 0.9025 - val_accuracy: 0.5851 - val_loss: 1.2909 - learning_rat
e: 6.2500e-05
Epoch 21/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 128ms/step - accuracy: 0.7097 - loss: 0.8727
Epoch 21: val_accuracy did not improve from 0.59104
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 151ms/step - accuracy: 0.7097 - loss: 0.8729 - val_accuracy: 0.5851 - val_loss: 1.2904 - learning_rat
e: 3.1250e-05
Epoch 22/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 127ms/step - accuracy: 0.7172 - loss: 0.8836
Epoch 22: val_accuracy did not improve from 0.59104

Epoch 22: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 149ms/step - accuracy: 0.7171 - loss: 0.8838 - val_accuracy: 0.5821 - val_loss: 1.2889 - learning_rat
e: 3.1250e-05
Epoch 23/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 128ms/step - accuracy: 0.7057 - loss: 0.8793
Epoch 23: val_accuracy did not improve from 0.59104
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 150ms/step - accuracy: 0.7058 - loss: 0.8794 - val_accuracy: 0.5821 - val_loss: 1.2871 - learning_rat
e: 1.5625e-05
Epoch 24/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 128ms/step - accuracy: 0.7071 - loss: 0.8712
Epoch 24: val_accuracy did not improve from 0.59104

Epoch 24: ReduceLROnPlateau reducing learning rate to 1e-05.
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 150ms/step - accuracy: 0.7070 - loss: 0.8713 - val_accuracy: 0.5821 - val_loss: 1.2873 - learning_rat
e: 1.5625e-05
```

```
Epoch 25/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 126ms/step - accuracy: 0.6928 - loss: 0.8947
Epoch 25: val_accuracy did not improve from 0.59104
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 170ms/step - accuracy: 0.6930 - loss: 0.8944 - val_accuracy: 0.5821 - val_loss: 1.2899 - learning_rat
e: 1.0000e-05
Epoch 26/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 127ms/step - accuracy: 0.7062 - loss: 0.9006
Epoch 26: val_accuracy did not improve from 0.59104
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 149ms/step - accuracy: 0.7063 - loss: 0.9003 - val_accuracy: 0.5851 - val_loss: 1.2876 - learning_rat
e: 1.0000e-05
Epoch 27/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 128ms/step - accuracy: 0.7025 - loss: 0.8712
Epoch 27: val_accuracy did not improve from 0.59104
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 150ms/step - accuracy: 0.7025 - loss: 0.8713 - val_accuracy: 0.5821 - val_loss: 1.2892 - learning_rat
e: 1.0000e-05
Epoch 28/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 129ms/step - accuracy: 0.6949 - loss: 0.8984
Epoch 28: val_accuracy did not improve from 0.59104
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 151ms/step - accuracy: 0.6949 - loss: 0.8981 - val_accuracy: 0.5851 - val_loss: 1.2901 - learning_rat
e: 1.0000e-05
Epoch 29/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 127ms/step - accuracy: 0.7347 - loss: 0.8444
Epoch 29: val_accuracy did not improve from 0.59104
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 149ms/step - accuracy: 0.7343 - loss: 0.8449 - val_accuracy: 0.5851 - val_loss: 1.2900 - learning_rat
e: 1.0000e-05
Epoch 30/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 127ms/step - accuracy: 0.7205 - loss: 0.8515
Epoch 30: val_accuracy did not improve from 0.59104
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 150ms/step - accuracy: 0.7204 - loss: 0.8517 - val_accuracy: 0.5851 - val_loss: 1.2908 - learning_rat
e: 1.0000e-05
Epoch 31/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 129ms/step - accuracy: 0.6850 - loss: 0.9143
Epoch 31: val_accuracy did not improve from 0.59104
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 151ms/step - accuracy: 0.6852 - loss: 0.9141 - val_accuracy: 0.5851 - val_loss: 1.2890 - learning_rat
e: 1.0000e-05
Epoch 32/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 128ms/step - accuracy: 0.6955 - loss: 0.8821
Epoch 32: val_accuracy did not improve from 0.59104
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 171ms/step - accuracy: 0.6956 - loss: 0.8821 - val_accuracy: 0.5851 - val_loss: 1.2894 - learning_rat
e: 1.0000e-05
Epoch 33/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 127ms/step - accuracy: 0.6992 - loss: 0.8883
Epoch 33: val_accuracy did not improve from 0.59104
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 149ms/step - accuracy: 0.6993 - loss: 0.8882 - val_accuracy: 0.5881 - val_loss: 1.2896 - learning_rat
e: 1.0000e-05
Epoch 34/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 128ms/step - accuracy: 0.7210 - loss: 0.8732
```

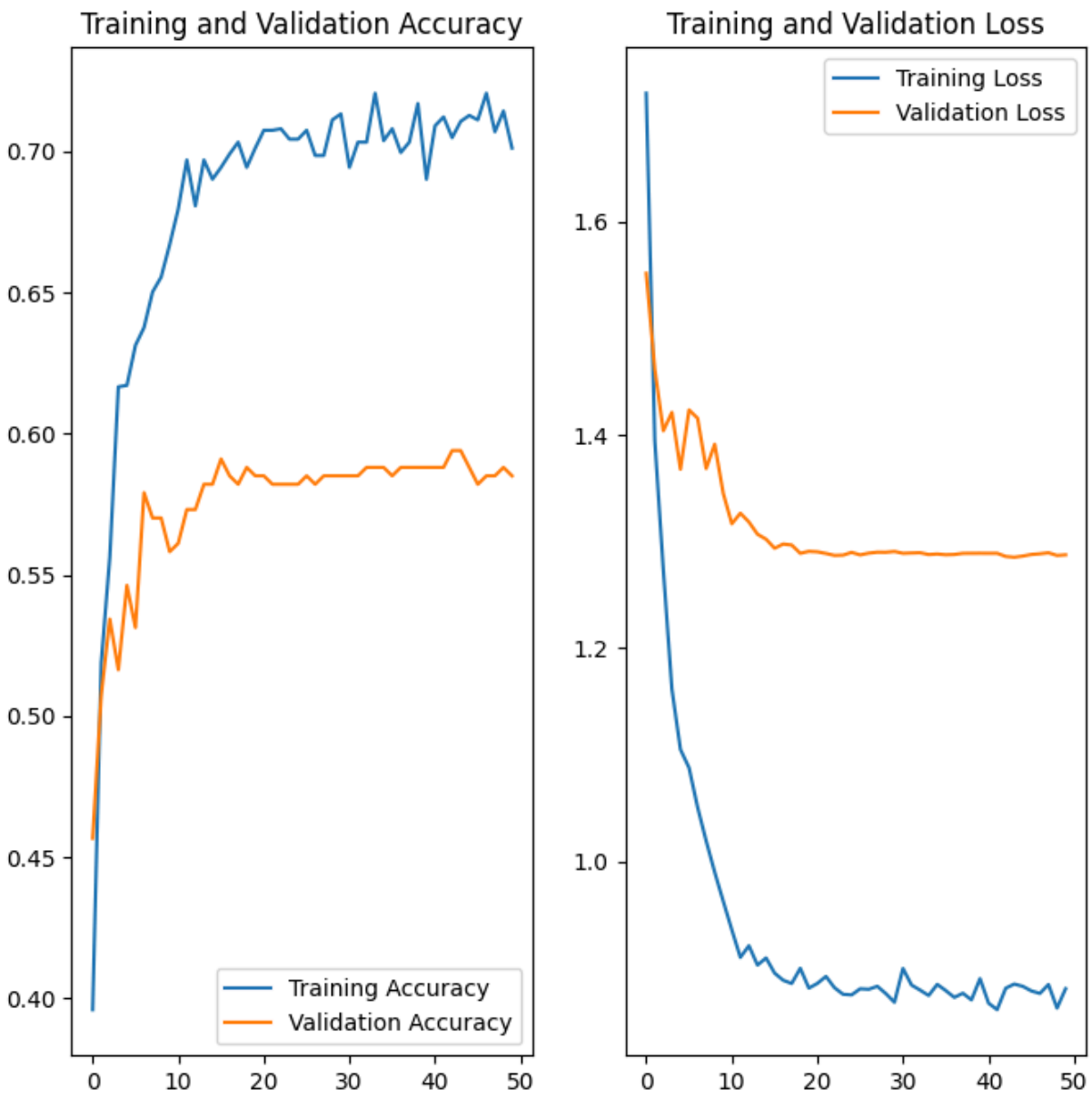
```
Epoch 34: val_accuracy did not improve from 0.59104
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 150ms/step - accuracy: 0.7210 - loss: 0.8732 - val_accuracy: 0.5881 - val_loss: 1.2879 - learning_rat
e: 1.0000e-05
Epoch 35/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 129ms/step - accuracy: 0.7025 - loss: 0.8662
Epoch 35: val_accuracy did not improve from 0.59104
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 151ms/step - accuracy: 0.7025 - loss: 0.8665 - val_accuracy: 0.5881 - val_loss: 1.2884 - learning_rat
e: 1.0000e-05
Epoch 36/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 129ms/step - accuracy: 0.7109 - loss: 0.8652
Epoch 36: val_accuracy did not improve from 0.59104
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 152ms/step - accuracy: 0.7108 - loss: 0.8655 - val_accuracy: 0.5851 - val_loss: 1.2877 - learning_rat
e: 1.0000e-05
Epoch 37/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 129ms/step - accuracy: 0.7153 - loss: 0.8490
Epoch 37: val_accuracy did not improve from 0.59104
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 151ms/step - accuracy: 0.7150 - loss: 0.8494 - val_accuracy: 0.5881 - val_loss: 1.2880 - learning_rat
e: 1.0000e-05
Epoch 38/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 129ms/step - accuracy: 0.6883 - loss: 0.8978
Epoch 38: val_accuracy did not improve from 0.59104
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 151ms/step - accuracy: 0.6886 - loss: 0.8974 - val_accuracy: 0.5881 - val_loss: 1.2890 - learning_rat
e: 1.0000e-05
Epoch 39/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step - accuracy: 0.7149 - loss: 0.8866
Epoch 39: val_accuracy did not improve from 0.59104
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 152ms/step - accuracy: 0.7149 - loss: 0.8863 - val_accuracy: 0.5881 - val_loss: 1.2891 - learning_rat
e: 1.0000e-05
Epoch 40/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 129ms/step - accuracy: 0.6866 - loss: 0.8958
Epoch 40: val_accuracy did not improve from 0.59104
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 151ms/step - accuracy: 0.6866 - loss: 0.8957 - val_accuracy: 0.5881 - val_loss: 1.2891 - learning_rat
e: 1.0000e-05
Epoch 41/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 129ms/step - accuracy: 0.7004 - loss: 0.8929
Epoch 41: val_accuracy did not improve from 0.59104
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 151ms/step - accuracy: 0.7005 - loss: 0.8925 - val_accuracy: 0.5881 - val_loss: 1.2891 - learning_rat
e: 1.0000e-05
Epoch 42/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step - accuracy: 0.7182 - loss: 0.8415
Epoch 42: val_accuracy did not improve from 0.59104
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 152ms/step - accuracy: 0.7181 - loss: 0.8418 - val_accuracy: 0.5881 - val_loss: 1.2891 - learning_rat
e: 1.0000e-05
Epoch 43/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 129ms/step - accuracy: 0.7199 - loss: 0.8596
Epoch 43: val_accuracy improved from 0.59104 to 0.59403, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_augmente
d.keras
```

```
60/60 ━━━━━━━━━━━━━━━━━━━━ 11s 187ms/step - accuracy: 0.7196 - loss: 0.8599 - val_accuracy: 0.5940 - val_loss: 1.2861 - learning_rat
e: 1.0000e-05
Epoch 44/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 128ms/step - accuracy: 0.6981 - loss: 0.9016
Epoch 44: val_accuracy did not improve from 0.59403
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 151ms/step - accuracy: 0.6983 - loss: 0.9014 - val_accuracy: 0.5940 - val_loss: 1.2854 - learning_rat
e: 1.0000e-05
Epoch 45/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step - accuracy: 0.7200 - loss: 0.8715
Epoch 45: val_accuracy did not improve from 0.59403
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 152ms/step - accuracy: 0.7199 - loss: 0.8717 - val_accuracy: 0.5881 - val_loss: 1.2865 - learning_rat
e: 1.0000e-05
Epoch 46/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step - accuracy: 0.7237 - loss: 0.8730
Epoch 46: val_accuracy did not improve from 0.59403
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 152ms/step - accuracy: 0.7235 - loss: 0.8731 - val_accuracy: 0.5821 - val_loss: 1.2880 - learning_rat
e: 1.0000e-05
Epoch 47/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 129ms/step - accuracy: 0.7072 - loss: 0.9049
Epoch 47: val_accuracy did not improve from 0.59403
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 152ms/step - accuracy: 0.7074 - loss: 0.9044 - val_accuracy: 0.5851 - val_loss: 1.2886 - learning_rat
e: 1.0000e-05
Epoch 48/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 128ms/step - accuracy: 0.7205 - loss: 0.8759
Epoch 48: val_accuracy did not improve from 0.59403
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 151ms/step - accuracy: 0.7203 - loss: 0.8760 - val_accuracy: 0.5851 - val_loss: 1.2896 - learning_rat
e: 1.0000e-05
Epoch 49/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step - accuracy: 0.7102 - loss: 0.8751
Epoch 49: val_accuracy did not improve from 0.59403
60/60 ━━━━━━━━━━━━━━━━━━━━ 9s 152ms/step - accuracy: 0.7103 - loss: 0.8749 - val_accuracy: 0.5881 - val_loss: 1.2870 - learning_rat
e: 1.0000e-05
Epoch 50/50
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step - accuracy: 0.6989 - loss: 0.8868
Epoch 50: val_accuracy did not improve from 0.59403
60/60 ━━━━━━━━━━━━━━━━━━━━ 10s 152ms/step - accuracy: 0.6989 - loss: 0.8867 - val_accuracy: 0.5851 - val_loss: 1.2876 - learning_rat
e: 1.0000e-05
```

Graficas del Segundo Modelo

In [34]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
epochs_range = range(EPOCHS)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



In [35]:
```
# Evaluar el modelo y obtener predicciones en el conjunto de prueba
y_true = np.concatenate([y for x, y in dataset_testing], axis=0)  # Etiquetas reales del conjunto de prueba
y_pred = model.predict(dataset_testing)  # Predicciones del modelo
y_pred = np.argmax(y_pred, axis=1)  # Convierte las probabilidades en etiquetas si es softmax
```

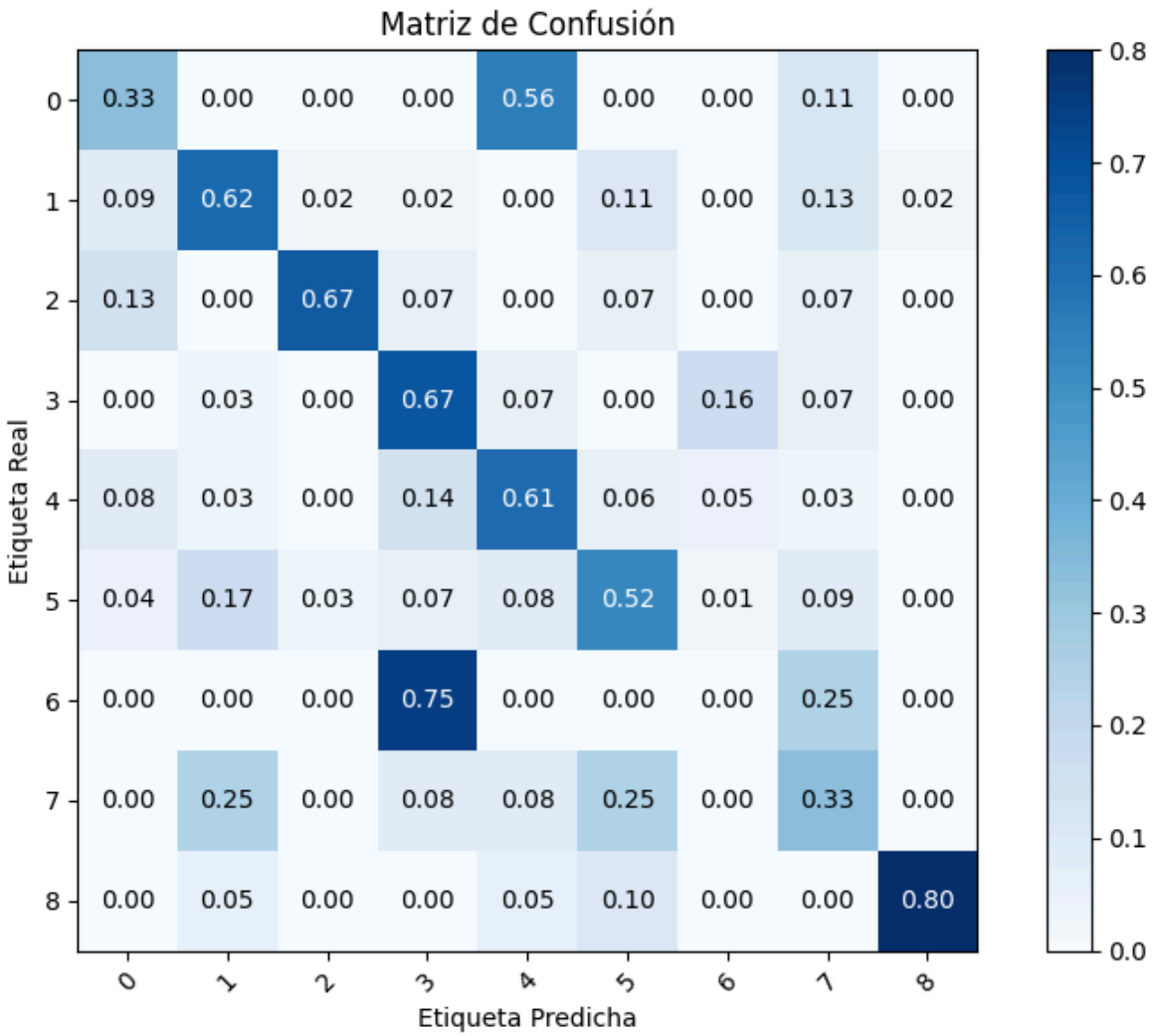
```
label_to_index = {label: idx for idx, label in enumerate(class_names)}
labels = list(label_to_index.values())  # Esto toma solo los valores numéricos del diccionario
labels
```

```
11/11 ━━━━━━━━━━━━━━━━━━━━ 4s 270ms/step
```

Out[35]: [0, 1, 2, 3, 4, 5, 6, 7, 8]

In [36]:

```
# Mostrar la matriz de confusión
mostrar_matriz_confusion(y_pred, y_true, labels=labels)  # Cambia los labels según tus clases
```



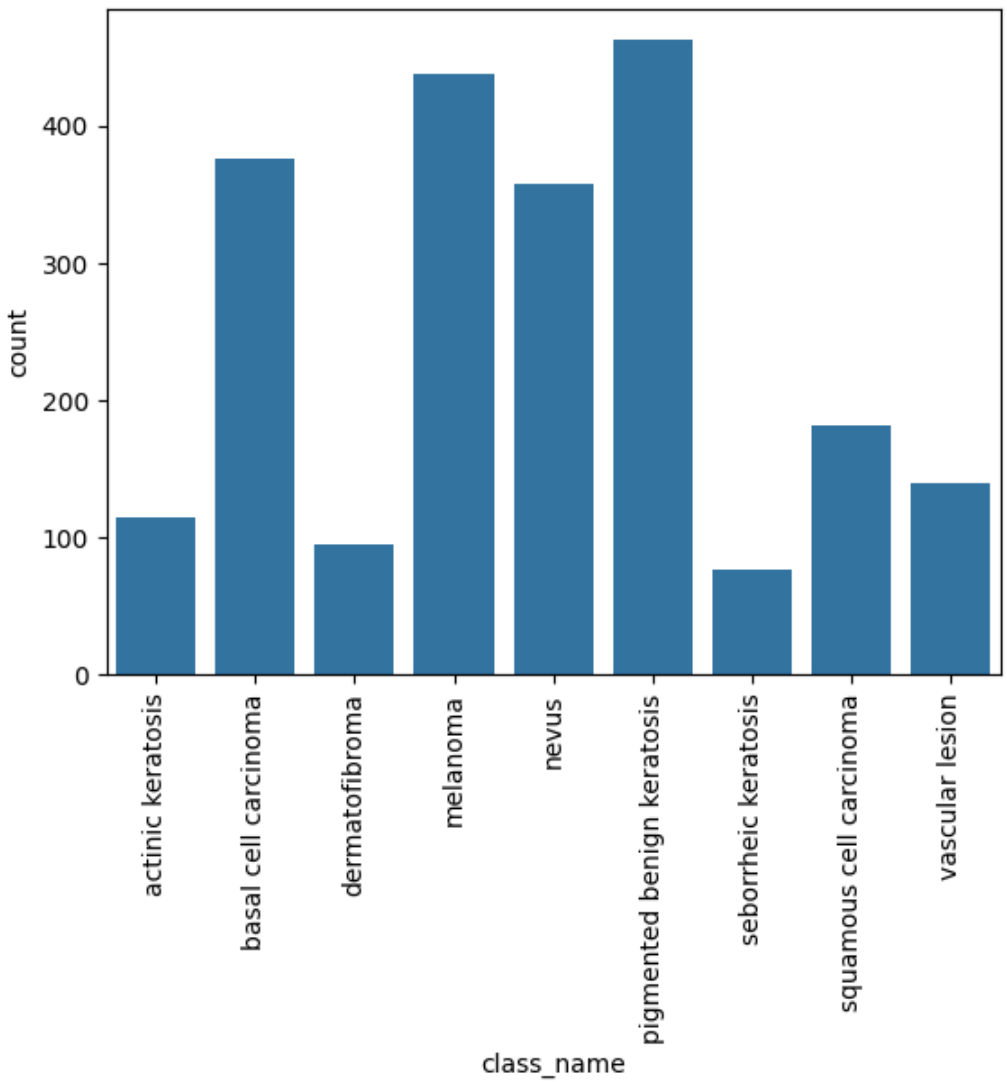
Resultados

- Después de aplicar técnicas de aumento de datos y mejorar la arquitectura del modelo, se observa una mayor estabilidad en su rendimiento: la precisión de entrenamiento y validación se mantienen cercanas, alrededor del 60%, lo que indica que el modelo está evitando el sobreajuste y generaliza mejor a datos nuevos.
- La pérdida también muestra una reducción rápida y se estabiliza en un valor bajo, con una diferencia pequeña entre el entrenamiento y la validación, lo cual sugiere que el modelo está aprendiendo patrones sin memorizar en exceso.

Analizando el Desbalance del Dataset

In [37]:
```
num_classes = len(class_names)
total = 0
all_count = []
class_name = []
for i in range(num_classes):
  count = len(list(data_dir_train.glob(class_names[i]+'/*.jpg')))
  total += count
print("total training image count = {} \n".format(total))
print("-------------------------------------")
for i in range(num_classes):
  count = len(list(data_dir_train.glob(class_names[i]+'/*.jpg')))
  print("Class name = ",class_names[i])
  print("count      = ",count)
  print("proportion = ",count/total)
  print("-------------------------------------")
  all_count.append(count)
  class_name.append(class_names[i])

temp_df = pd.DataFrame(list(zip(all_count, class_name)), columns = ['count', 'class_name'])
sns.barplot(data=temp_df, y="count", x="class_name")
plt.xticks(rotation=90)
plt.show()
```

```
total training image count = 2239

-------------------------------------
Class name =  actinic keratosis
count      =  114
proportion =  0.05091558731576597
-------------------------------------
Class name =  basal cell carcinoma
count      =  376
proportion =  0.16793211255024565
-------------------------------------
Class name =  dermatofibroma
count      =  95
proportion =  0.04242965609647164
-------------------------------------
Class name =  melanoma
count      =  438
proportion =  0.19562304600267977
-------------------------------------
Class name =  nevus
count      =  357
proportion =  0.15944618133095131
-------------------------------------
Class name =  pigmented benign keratosis
count      =  462
proportion =  0.20634211701652524
-------------------------------------
Class name =  seborrheic keratosis
count      =  77
proportion =  0.03439035283608754
-------------------------------------
Class name =  squamous cell carcinoma
count      =  181
proportion =  0.08083966056275123
-------------------------------------
Class name =  vascular lesion
count      =  139
proportion =  0.062081286288521664
-------------------------------------
```



Hallazgos

- La gráfica muestra que el conjunto de datos está desequilibrado, con algunas clases que tienen muchas más imágenes que otras. Por ejemplo, "nevus" y "pigmented benign keratosis" tienen más de 400 ejemplos cada una, mientras que otras clases, como "dermatofibroma" y "vascular lesion", tienen menos de 200.
- Este desequilibrio puede hacer que el modelo aprenda mejor a identificar las clases con más ejemplos y tenga más dificultad en reconocer las menos representadas.

- Para mejorar, podríamos usar técnicas como el aumento de datos en las clases con menos ejemplos o ajustar el modelo para que preste más atención a estas clases menos frecuentes.

Balanceo de Clases usando Augmentor

Utilizando Augmentor (https://augmentor.readthedocs.io/en/master/) para crear la distribución equitativa de la clase de 5000 imagenes por cada clase.

In [39]:
```
pip install Augmentor
```

```
Collecting Augmentor
  Downloading Augmentor-0.2.12-py2.py3-none-any.whl.metadata (1.3 kB)
Requirement already satisfied: Pillow>=5.2.0 in /usr/local/lib/python3.10/dist-packages (from Augmentor) (11.0.0)
Requirement already satisfied: tqdm>=4.9.0 in /usr/local/lib/python3.10/dist-packages (from Augmentor) (4.66.6)
Requirement already satisfied: numpy>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from Augmentor) (1.26.4)
Downloading Augmentor-0.2.12-py2.py3-none-any.whl (38 kB)
Installing collected packages: Augmentor
Successfully installed Augmentor-0.2.12
```

In [40]:
```
import Augmentor
import os

# Directorio de entrenamiento
path_to_training_dataset = '/content/drive/My Drive/IA/dataset/Train/'
output_base_dir = r'/content/drive/My Drive/IA/dataset/Augmented_Train'  # Directorio base para guardar las imágenes aumentadas

# Asegúrate de que el directorio de salida principal exista
os.makedirs(output_base_dir, exist_ok=True)

# Obtiene los nombres de las clases (carpetas dentro de Train)
class_names = os.listdir(path_to_training_dataset)

# Para cada clase en el conjunto de datos
for class_name in class_names:
    # Define la ruta de entrada y salida para cada clase
    input_class_path = os.path.join(path_to_training_dataset, class_name)
    output_class_path = os.path.join(output_base_dir, class_name)

    # Crear el directorio de salida de la clase si no existe
    os.makedirs(output_class_path, exist_ok=True)

    # Define el pipeline de Augmentor con la carpeta de entrada y salida
    p = Augmentor.Pipeline(source_directory=input_class_path, output_directory=output_class_path)
```

```
# Agregar operaciones de aumento de datos
p.rotate(probability=0.7, max_left_rotation=10, max_right_rotation=10)

# Generar imágenes aumentadas
p.sample(1000)
```

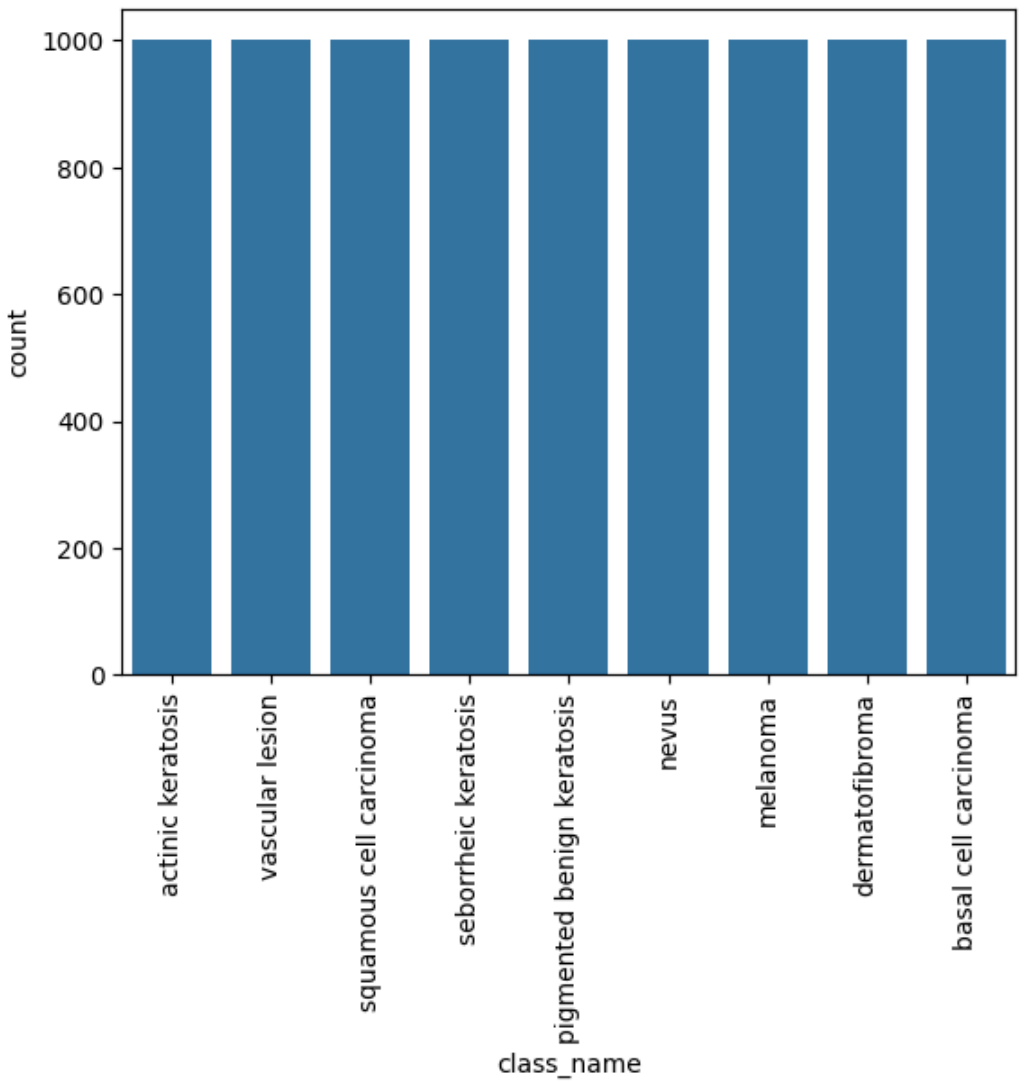
```
Initialised with 114 image(s) found.
Output directory set to /content/drive/My Drive/IA/dataset/Augmented_Train/actinic keratosis.
Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x7EEC4BA0FA90>: 100%|██████████| 1000/1000 [00:50<00:00, 19.77 Samples/s]
Initialised with 139 image(s) found.
Output directory set to /content/drive/My Drive/IA/dataset/Augmented_Train/vascular lesion.
Processing <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=600x450 at 0x7EEBB0203E50>: 100%|██████████| 1000/1000 [00:40<00:00, 24.88 Samples/s]
Initialised with 181 image(s) found.
Output directory set to /content/drive/My Drive/IA/dataset/Augmented_Train/squamous cell carcinoma.
Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x7EEC464E0A90>: 100%|██████████| 1000/1000 [00:41<00:00, 24.34 Samples/s]
Initialised with 77 image(s) found.
Output directory set to /content/drive/My Drive/IA/dataset/Augmented_Train/seborrheic keratosis.
Processing <PIL.Image.Image image mode=RGB size=1024x768 at 0x7EEC464E1A20>: 100%|██████████| 1000/1000 [01:22<00:00, 12.11 Samples/s]
Initialised with 462 image(s) found.
Output directory set to /content/drive/My Drive/IA/dataset/Augmented_Train/pigmented benign keratosis.
Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x7EEC4BA0F850>: 100%|██████████| 1000/1000 [00:40<00:00, 24.63 Samples/s]
Initialised with 357 image(s) found.
Output directory set to /content/drive/My Drive/IA/dataset/Augmented_Train/nevus.
Processing <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=767x576 at 0x7EEC4BF553F0>: 100%|██████████| 1000/1000 [02:32<00:00,  6.57 Samples/s]
Initialised with 438 image(s) found.
Output directory set to /content/drive/My Drive/IA/dataset/Augmented_Train/melanoma.
Processing <PIL.Image.Image image mode=RGB size=1024x768 at 0x7EEC4BBDFC40>: 100%|██████████| 1000/1000 [02:48<00:00,  5.94 Samples/s]
Initialised with 95 image(s) found.
Output directory set to /content/drive/My Drive/IA/dataset/Augmented_Train/dermatofibroma.
Processing <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=600x450 at 0x7EEBB069D960>: 100%|██████████| 1000/1000 [00:38<00:00, 25.69 Samples/s]
Initialised with 376 image(s) found.
Output directory set to /content/drive/My Drive/IA/dataset/Augmented_Train/basal cell carcinoma.
Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x7EEC463E2B90>: 100%|██████████| 1000/1000 [00:41<00:00, 24.13 Samples/s]
```

In [41]:
```python
# Ruta al directorio con las imágenes aumentadas
output_dir = pathlib.Path('/content/drive/My Drive/IA/dataset/Augmented_Train/')

# Contar todas las imágenes .jpg dentro de cada subcarpeta de clase
image_count_train = len(list(output_dir.glob('*/*.jpg')))
print("Total de imágenes de entrenamiento:", image_count_train)
```

```
Total de imágenes de entrenamiento: 9000
```

In [42]:
```python
# Ruta al directorio con imágenes aumentadas
output_dir = pathlib.Path('/content/drive/My Drive/IA/dataset/Augmented_Train')

# Obtener el número de clases
class_names = os.listdir(output_dir)
num_classes = len(class_names)

# Variables para contar las imágenes y almacenar resultados
all_count = []
class_name_list = []

# Calcular el número de imágenes por clase
for class_name in class_names:
    # Contar las imágenes en cada carpeta de clase
    count = len(list(output_dir.glob(f"{class_name}/*.jpg")))
    all_count.append(count)
    class_name_list.append(class_name)

# Calcular el total después de contar todas las clases
total = sum(all_count)

# Imprimir el conteo y la proporción de imágenes por clase
for i, class_name in enumerate(class_name_list):
    count = all_count[i]
    proportion = count / total
    print(f"Class name = {class_name}")
    print(f"Count      = {count}")
    print(f"Proportion = {proportion:.2%}")
    print("-------------------------------------")

# Crear un DataFrame para la visualización
temp_df = pd.DataFrame(list(zip(all_count, class_name_list)), columns=['count', 'class_name'])

# Graficar el conteo de imágenes por clase
sns.barplot(data=temp_df, y="count", x="class_name")
plt.xticks(rotation=90)
plt.show()
```

```
Class name = actinic keratosis
Count      = 1000
Proportion = 11.11%
-------------------------------------
Class name = vascular lesion
Count      = 1000
Proportion = 11.11%
-------------------------------------
Class name = squamous cell carcinoma
Count      = 1000
Proportion = 11.11%
-------------------------------------
Class name = seborrheic keratosis
Count      = 1000
Proportion = 11.11%
-------------------------------------
Class name = pigmented benign keratosis
Count      = 1000
Proportion = 11.11%
-------------------------------------
Class name = nevus
Count      = 1000
Proportion = 11.11%
-------------------------------------
Class name = melanoma
Count      = 1000
Proportion = 11.11%
-------------------------------------
Class name = dermatofibroma
Count      = 1000
Proportion = 11.11%
-------------------------------------
Class name = basal cell carcinoma
Count      = 1000
Proportion = 11.11%
-------------------------------------
```



Resultados

- Balanceamos las clases a 2000 imagenes

Modelo 3 - Balanceado con imagenes de augmentor

In [43]:
```
dataset_training = tf.keras.preprocessing.image_dataset_from_directory(
  output_dir,
  seed=123,
  validation_split = 0.15,
  subset = 'training',
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
Found 9000 files belonging to 9 classes.
Using 7650 files for training.
```

In [44]:
```
dataset_testing = tf.keras.preprocessing.image_dataset_from_directory(
  output_dir,
  seed=123,
  validation_split = 0.15,
  subset = 'validation',
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
Found 9000 files belonging to 9 classes.
Using 1350 files for validation.
```

In [45]:
```
print(dataset_training.class_names)
```

```
['actinic keratosis', 'basal cell carcinoma', 'dermatofibroma', 'melanoma', 'nevus', 'pigmented benign keratosis', 'seborrheic kerat
osis', 'squamous cell carcinoma', 'vascular lesion']
```

In [46]:
```
num_classes = 9
model = tf.keras.Sequential([
    tf.keras.layers.InputLayer(input_shape=(img_height, img_width, 3)),
    pretrained,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(num_classes, activation='softmax')
])
```

In [47]:
```
# Definir el optimizador con el argumento correcto
opt = Adam(learning_rate=0.001)

# Compilar el modelo
model.compile(
    optimizer=opt,
    loss=SparseCategoricalCrossentropy(from_logits=True),
    metrics=['accuracy']
)
```

In [48]:
```
lr_reduce = ReduceLROnPlateau(monitor='val_accuracy', factor=0.5, patience=2,mode='max', min_lr=0.00001,verbose=1)
early_stop = EarlyStopping(monitor="val_loss", patience=2, verbose=1)
```

```
model_chkpt = ModelCheckpoint('/content/drive/My Drive/IA/best_skin_cancer_model_balanced.keras',save_best_only=True, monitor='val_a

callback_list = [model_chkpt,lr_reduce]
```

In [49]:

```
history = model.fit(
    dataset_training,
    epochs=EPOCHS,
    batch_size=batch_size,
    validation_data=dataset_testing,
    callbacks=callback_list
)
```

```
Epoch 1/50
240/240 ━━━━━━━━━━━━━━━━━━━━ 0s 197ms/step - accuracy: 0.4025 - loss: 1.6755
Epoch 1: val_accuracy improved from -inf to 0.65852, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_balanced.kera
s
240/240 ━━━━━━━━━━━━━━━━━━━━ 74s 261ms/step - accuracy: 0.4029 - loss: 1.6743 - val_accuracy: 0.6585 - val_loss: 1.0309 - learning_r
ate: 0.0010
Epoch 2/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 182ms/step - accuracy: 0.6772 - loss: 0.9524
Epoch 2: val_accuracy improved from 0.65852 to 0.69556, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_balanced.k
eras
240/240 ━━━━━━━━━━━━━━━━━━━━ 71s 232ms/step - accuracy: 0.6773 - loss: 0.9521 - val_accuracy: 0.6956 - val_loss: 0.8577 - learning_r
ate: 0.0010
Epoch 3/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 173ms/step - accuracy: 0.7529 - loss: 0.7664
Epoch 3: val_accuracy improved from 0.69556 to 0.74296, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_balanced.k
eras
240/240 ━━━━━━━━━━━━━━━━━━━━ 75s 205ms/step - accuracy: 0.7529 - loss: 0.7663 - val_accuracy: 0.7430 - val_loss: 0.7516 - learning_r
ate: 0.0010
Epoch 4/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 172ms/step - accuracy: 0.7878 - loss: 0.6601
Epoch 4: val_accuracy improved from 0.74296 to 0.78000, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_balanced.k
eras
240/240 ━━━━━━━━━━━━━━━━━━━━ 82s 204ms/step - accuracy: 0.7878 - loss: 0.6601 - val_accuracy: 0.7800 - val_loss: 0.6838 - learning_r
ate: 0.0010
Epoch 5/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 180ms/step - accuracy: 0.8266 - loss: 0.5742
Epoch 5: val_accuracy improved from 0.78000 to 0.79481, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_balanced.k
eras
240/240 ━━━━━━━━━━━━━━━━━━━━ 88s 227ms/step - accuracy: 0.8266 - loss: 0.5742 - val_accuracy: 0.7948 - val_loss: 0.6336 - learning_r
ate: 0.0010
Epoch 6/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 176ms/step - accuracy: 0.8363 - loss: 0.5231
Epoch 6: val_accuracy improved from 0.79481 to 0.80444, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_balanced.k
eras
240/240 ━━━━━━━━━━━━━━━━━━━━ 78s 212ms/step - accuracy: 0.8364 - loss: 0.5230 - val_accuracy: 0.8044 - val_loss: 0.6024 - learning_r
ate: 0.0010
Epoch 7/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 173ms/step - accuracy: 0.8554 - loss: 0.4871
Epoch 7: val_accuracy improved from 0.80444 to 0.82667, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_balanced.k
eras
240/240 ━━━━━━━━━━━━━━━━━━━━ 84s 221ms/step - accuracy: 0.8555 - loss: 0.4870 - val_accuracy: 0.8267 - val_loss: 0.5568 - learning_r
ate: 0.0010
Epoch 8/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 174ms/step - accuracy: 0.8706 - loss: 0.4415
Epoch 8: val_accuracy improved from 0.82667 to 0.83185, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_balanced.k
eras
240/240 ━━━━━━━━━━━━━━━━━━━━ 54s 223ms/step - accuracy: 0.8706 - loss: 0.4415 - val_accuracy: 0.8319 - val_loss: 0.5326 - learning_r
```

```
ate: 0.0010
Epoch 9/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 173ms/step - accuracy: 0.8911 - loss: 0.4047
Epoch 9: val_accuracy improved from 0.83185 to 0.84370, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_balanced.k
eras
240/240 ━━━━━━━━━━━━━━━━━━━━ 78s 207ms/step - accuracy: 0.8911 - loss: 0.4047 - val_accuracy: 0.8437 - val_loss: 0.5080 - learning_r
ate: 0.0010
Epoch 10/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 176ms/step - accuracy: 0.8988 - loss: 0.3701
Epoch 10: val_accuracy improved from 0.84370 to 0.85333, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_balanced.
keras
240/240 ━━━━━━━━━━━━━━━━━━━━ 54s 225ms/step - accuracy: 0.8988 - loss: 0.3702 - val_accuracy: 0.8533 - val_loss: 0.4940 - learning_r
ate: 0.0010
Epoch 11/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 176ms/step - accuracy: 0.9076 - loss: 0.3547
Epoch 11: val_accuracy improved from 0.85333 to 0.85852, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_balanced.
keras
240/240 ━━━━━━━━━━━━━━━━━━━━ 50s 209ms/step - accuracy: 0.9076 - loss: 0.3547 - val_accuracy: 0.8585 - val_loss: 0.4680 - learning_r
ate: 0.0010
Epoch 12/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 172ms/step - accuracy: 0.9136 - loss: 0.3347
Epoch 12: val_accuracy did not improve from 0.85852
240/240 ━━━━━━━━━━━━━━━━━━━━ 83s 214ms/step - accuracy: 0.9136 - loss: 0.3347 - val_accuracy: 0.8578 - val_loss: 0.4634 - learning_r
ate: 0.0010
Epoch 13/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 171ms/step - accuracy: 0.9171 - loss: 0.3200
Epoch 13: val_accuracy improved from 0.85852 to 0.86593, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_balanced.
keras
240/240 ━━━━━━━━━━━━━━━━━━━━ 83s 219ms/step - accuracy: 0.9171 - loss: 0.3200 - val_accuracy: 0.8659 - val_loss: 0.4370 - learning_r
ate: 0.0010
Epoch 14/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 170ms/step - accuracy: 0.9220 - loss: 0.2992
Epoch 14: val_accuracy did not improve from 0.86593
240/240 ━━━━━━━━━━━━━━━━━━━━ 80s 212ms/step - accuracy: 0.9220 - loss: 0.2993 - val_accuracy: 0.8644 - val_loss: 0.4280 - learning_r
ate: 0.0010
Epoch 15/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 172ms/step - accuracy: 0.9248 - loss: 0.2813
Epoch 15: val_accuracy did not improve from 0.86593

Epoch 15: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
240/240 ━━━━━━━━━━━━━━━━━━━━ 80s 205ms/step - accuracy: 0.9248 - loss: 0.2813 - val_accuracy: 0.8533 - val_loss: 0.4382 - learning_r
ate: 0.0010
Epoch 16/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 172ms/step - accuracy: 0.9364 - loss: 0.2579
Epoch 16: val_accuracy improved from 0.86593 to 0.87704, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_balanced.
keras
240/240 ━━━━━━━━━━━━━━━━━━━━ 83s 209ms/step - accuracy: 0.9364 - loss: 0.2579 - val_accuracy: 0.8770 - val_loss: 0.4051 - learning_r
```

```
ate: 5.0000e-04
Epoch 17/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 174ms/step - accuracy: 0.9406 - loss: 0.2504
Epoch 17: val_accuracy did not improve from 0.87704
240/240 ━━━━━━━━━━━━━━━━━━━━ 84s 216ms/step - accuracy: 0.9405 - loss: 0.2504 - val_accuracy: 0.8741 - val_loss: 0.4005 - learning_r
ate: 5.0000e-04
Epoch 18/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 175ms/step - accuracy: 0.9330 - loss: 0.2554
Epoch 18: val_accuracy improved from 0.87704 to 0.88815, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_balanced.
keras
240/240 ━━━━━━━━━━━━━━━━━━━━ 84s 224ms/step - accuracy: 0.9330 - loss: 0.2554 - val_accuracy: 0.8881 - val_loss: 0.3944 - learning_r
ate: 5.0000e-04
Epoch 19/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 175ms/step - accuracy: 0.9432 - loss: 0.2382
Epoch 19: val_accuracy did not improve from 0.88815
240/240 ━━━━━━━━━━━━━━━━━━━━ 52s 217ms/step - accuracy: 0.9432 - loss: 0.2383 - val_accuracy: 0.8756 - val_loss: 0.3891 - learning_r
ate: 5.0000e-04
Epoch 20/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 171ms/step - accuracy: 0.9414 - loss: 0.2341
Epoch 20: val_accuracy did not improve from 0.88815

Epoch 20: ReduceLROnPlateau reducing learning rate to 0.00025000000118743628.
240/240 ━━━━━━━━━━━━━━━━━━━━ 81s 213ms/step - accuracy: 0.9413 - loss: 0.2342 - val_accuracy: 0.8830 - val_loss: 0.3854 - learning_r
ate: 5.0000e-04
Epoch 21/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 172ms/step - accuracy: 0.9470 - loss: 0.2269
Epoch 21: val_accuracy did not improve from 0.88815
240/240 ━━━━━━━━━━━━━━━━━━━━ 52s 214ms/step - accuracy: 0.9470 - loss: 0.2269 - val_accuracy: 0.8874 - val_loss: 0.3782 - learning_r
ate: 2.5000e-04
Epoch 22/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 174ms/step - accuracy: 0.9457 - loss: 0.2224
Epoch 22: val_accuracy did not improve from 0.88815

Epoch 22: ReduceLROnPlateau reducing learning rate to 0.00012500000059371814.
240/240 ━━━━━━━━━━━━━━━━━━━━ 80s 206ms/step - accuracy: 0.9457 - loss: 0.2224 - val_accuracy: 0.8874 - val_loss: 0.3752 - learning_r
ate: 2.5000e-04
Epoch 23/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 174ms/step - accuracy: 0.9508 - loss: 0.2166
Epoch 23: val_accuracy did not improve from 0.88815
240/240 ━━━━━━━━━━━━━━━━━━━━ 84s 216ms/step - accuracy: 0.9508 - loss: 0.2166 - val_accuracy: 0.8881 - val_loss: 0.3720 - learning_r
ate: 1.2500e-04
Epoch 24/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 171ms/step - accuracy: 0.9485 - loss: 0.2146
Epoch 24: val_accuracy improved from 0.88815 to 0.88889, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_balanced.
keras
240/240 ━━━━━━━━━━━━━━━━━━━━ 82s 218ms/step - accuracy: 0.9485 - loss: 0.2146 - val_accuracy: 0.8889 - val_loss: 0.3713 - learning_r
ate: 1.2500e-04
```

```
Epoch 25/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 172ms/step - accuracy: 0.9516 - loss: 0.2144
Epoch 25: val_accuracy improved from 0.88889 to 0.88963, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_balanced.
keras
240/240 ━━━━━━━━━━━━━━━━━━━━ 82s 220ms/step - accuracy: 0.9516 - loss: 0.2144 - val_accuracy: 0.8896 - val_loss: 0.3696 - learning_r
ate: 1.2500e-04
Epoch 26/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 171ms/step - accuracy: 0.9479 - loss: 0.2139
Epoch 26: val_accuracy did not improve from 0.88963
240/240 ━━━━━━━━━━━━━━━━━━━━ 78s 203ms/step - accuracy: 0.9479 - loss: 0.2139 - val_accuracy: 0.8896 - val_loss: 0.3694 - learning_r
ate: 1.2500e-04
Epoch 27/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 171ms/step - accuracy: 0.9520 - loss: 0.2071
Epoch 27: val_accuracy did not improve from 0.88963

Epoch 27: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
240/240 ━━━━━━━━━━━━━━━━━━━━ 51s 214ms/step - accuracy: 0.9520 - loss: 0.2072 - val_accuracy: 0.8896 - val_loss: 0.3676 - learning_r
ate: 1.2500e-04
Epoch 28/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 170ms/step - accuracy: 0.9499 - loss: 0.2083
Epoch 28: val_accuracy did not improve from 0.88963
240/240 ━━━━━━━━━━━━━━━━━━━━ 82s 212ms/step - accuracy: 0.9498 - loss: 0.2083 - val_accuracy: 0.8896 - val_loss: 0.3656 - learning_r
ate: 6.2500e-05
Epoch 29/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 171ms/step - accuracy: 0.9524 - loss: 0.2046
Epoch 29: val_accuracy improved from 0.88963 to 0.89111, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_balanced.
keras
240/240 ━━━━━━━━━━━━━━━━━━━━ 84s 221ms/step - accuracy: 0.9524 - loss: 0.2046 - val_accuracy: 0.8911 - val_loss: 0.3648 - learning_r
ate: 6.2500e-05
Epoch 30/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 173ms/step - accuracy: 0.9524 - loss: 0.2044
Epoch 30: val_accuracy improved from 0.89111 to 0.89259, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_balanced.
keras
240/240 ━━━━━━━━━━━━━━━━━━━━ 80s 211ms/step - accuracy: 0.9523 - loss: 0.2044 - val_accuracy: 0.8926 - val_loss: 0.3653 - learning_r
ate: 6.2500e-05
Epoch 31/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 177ms/step - accuracy: 0.9511 - loss: 0.2042
Epoch 31: val_accuracy did not improve from 0.89259
240/240 ━━━━━━━━━━━━━━━━━━━━ 49s 203ms/step - accuracy: 0.9511 - loss: 0.2043 - val_accuracy: 0.8911 - val_loss: 0.3632 - learning_r
ate: 6.2500e-05
Epoch 32/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 175ms/step - accuracy: 0.9532 - loss: 0.2035
Epoch 32: val_accuracy did not improve from 0.89259

Epoch 32: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
240/240 ━━━━━━━━━━━━━━━━━━━━ 85s 217ms/step - accuracy: 0.9532 - loss: 0.2036 - val_accuracy: 0.8919 - val_loss: 0.3635 - learning_r
ate: 6.2500e-05
```

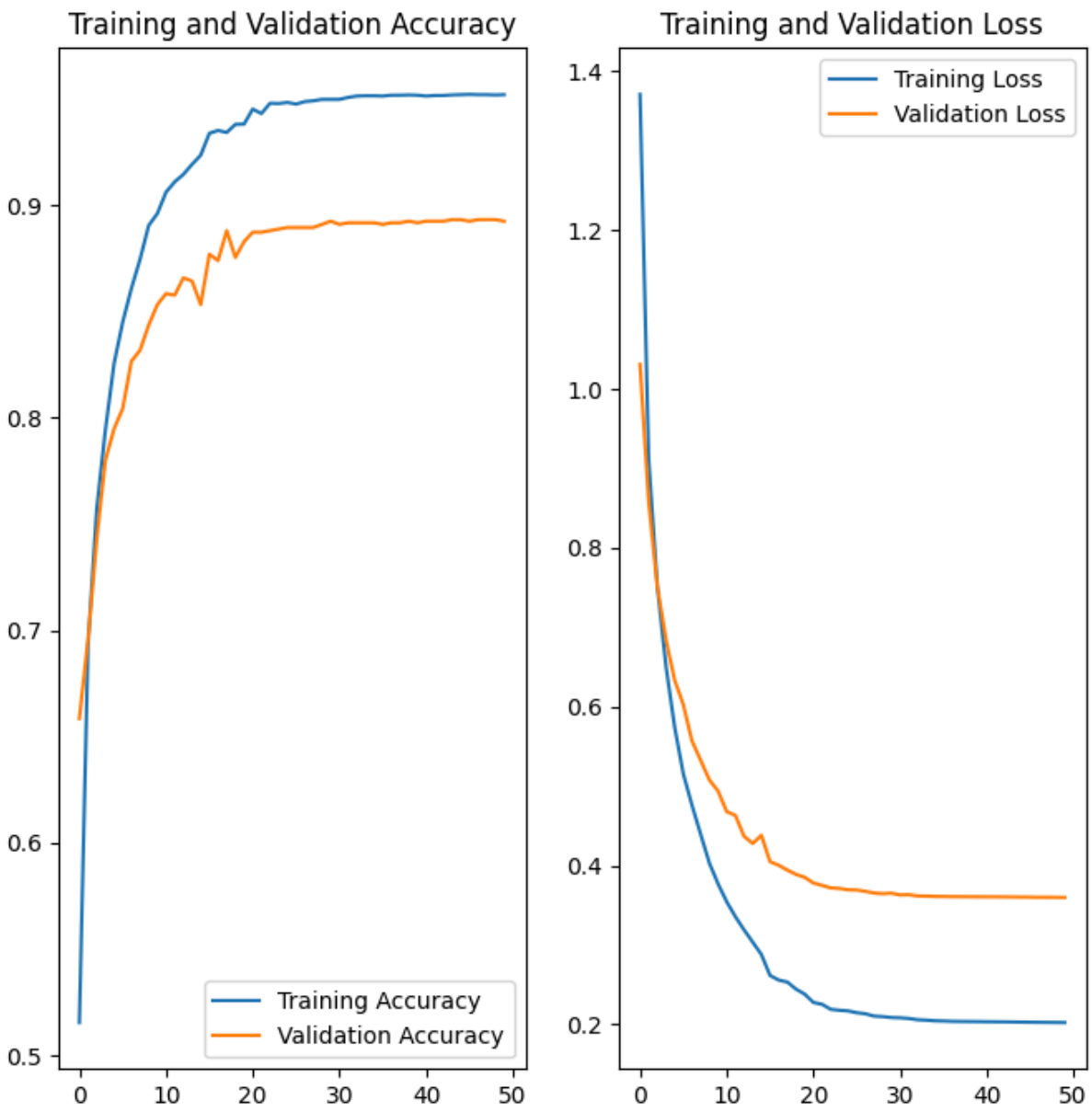
```
Epoch 33/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 172ms/step - accuracy: 0.9525 - loss: 0.2014
Epoch 33: val_accuracy did not improve from 0.89259
240/240 ━━━━━━━━━━━━━━━━━━━━ 81s 214ms/step - accuracy: 0.9525 - loss: 0.2015 - val_accuracy: 0.8919 - val_loss: 0.3619 - learning_r
ate: 3.1250e-05
Epoch 34/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 171ms/step - accuracy: 0.9526 - loss: 0.2009
Epoch 34: val_accuracy did not improve from 0.89259

Epoch 34: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
240/240 ━━━━━━━━━━━━━━━━━━━━ 79s 203ms/step - accuracy: 0.9526 - loss: 0.2010 - val_accuracy: 0.8919 - val_loss: 0.3617 - learning_r
ate: 3.1250e-05
Epoch 35/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 170ms/step - accuracy: 0.9531 - loss: 0.2025
Epoch 35: val_accuracy did not improve from 0.89259
240/240 ━━━━━━━━━━━━━━━━━━━━ 84s 212ms/step - accuracy: 0.9531 - loss: 0.2025 - val_accuracy: 0.8919 - val_loss: 0.3613 - learning_r
ate: 1.5625e-05
Epoch 36/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 169ms/step - accuracy: 0.9542 - loss: 0.1981
Epoch 36: val_accuracy did not improve from 0.89259

Epoch 36: ReduceLROnPlateau reducing learning rate to 1e-05.
240/240 ━━━━━━━━━━━━━━━━━━━━ 80s 202ms/step - accuracy: 0.9542 - loss: 0.1981 - val_accuracy: 0.8911 - val_loss: 0.3612 - learning_r
ate: 1.5625e-05
Epoch 37/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 178ms/step - accuracy: 0.9525 - loss: 0.2028
Epoch 37: val_accuracy did not improve from 0.89259
240/240 ━━━━━━━━━━━━━━━━━━━━ 49s 205ms/step - accuracy: 0.9525 - loss: 0.2029 - val_accuracy: 0.8919 - val_loss: 0.3611 - learning_r
ate: 1.0000e-05
Epoch 38/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 178ms/step - accuracy: 0.9551 - loss: 0.1973
Epoch 38: val_accuracy did not improve from 0.89259
240/240 ━━━━━━━━━━━━━━━━━━━━ 83s 207ms/step - accuracy: 0.9550 - loss: 0.1974 - val_accuracy: 0.8919 - val_loss: 0.3610 - learning_r
ate: 1.0000e-05
Epoch 39/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 171ms/step - accuracy: 0.9529 - loss: 0.1999
Epoch 39: val_accuracy did not improve from 0.89259
240/240 ━━━━━━━━━━━━━━━━━━━━ 83s 213ms/step - accuracy: 0.9529 - loss: 0.2000 - val_accuracy: 0.8926 - val_loss: 0.3609 - learning_r
ate: 1.0000e-05
Epoch 40/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 171ms/step - accuracy: 0.9531 - loss: 0.1998
Epoch 40: val_accuracy did not improve from 0.89259
240/240 ━━━━━━━━━━━━━━━━━━━━ 79s 202ms/step - accuracy: 0.9531 - loss: 0.1998 - val_accuracy: 0.8919 - val_loss: 0.3609 - learning_r
ate: 1.0000e-05
Epoch 41/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 175ms/step - accuracy: 0.9533 - loss: 0.1986
Epoch 41: val_accuracy did not improve from 0.89259
```

```
240/240 ━━━━━━━━━━━━━━━━━━━━ 83s 207ms/step - accuracy: 0.9533 - loss: 0.1987 - val_accuracy: 0.8926 - val_loss: 0.3608 - learning_r
ate: 1.0000e-05
Epoch 42/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 172ms/step - accuracy: 0.9528 - loss: 0.2001
Epoch 42: val_accuracy did not improve from 0.89259
240/240 ━━━━━━━━━━━━━━━━━━━━ 84s 214ms/step - accuracy: 0.9528 - loss: 0.2001 - val_accuracy: 0.8926 - val_loss: 0.3608 - learning_r
ate: 1.0000e-05
Epoch 43/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 170ms/step - accuracy: 0.9551 - loss: 0.1963
Epoch 43: val_accuracy did not improve from 0.89259
240/240 ━━━━━━━━━━━━━━━━━━━━ 79s 202ms/step - accuracy: 0.9550 - loss: 0.1964 - val_accuracy: 0.8926 - val_loss: 0.3607 - learning_r
ate: 1.0000e-05
Epoch 44/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 170ms/step - accuracy: 0.9545 - loss: 0.1978
Epoch 44: val_accuracy improved from 0.89259 to 0.89333, saving model to /content/drive/My Drive/IA/best_skin_cancer_model_balanced.
keras
240/240 ━━━━━━━━━━━━━━━━━━━━ 86s 217ms/step - accuracy: 0.9544 - loss: 0.1978 - val_accuracy: 0.8933 - val_loss: 0.3606 - learning_r
ate: 1.0000e-05
Epoch 45/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 174ms/step - accuracy: 0.9539 - loss: 0.1987
Epoch 45: val_accuracy did not improve from 0.89333
240/240 ━━━━━━━━━━━━━━━━━━━━ 49s 205ms/step - accuracy: 0.9539 - loss: 0.1988 - val_accuracy: 0.8933 - val_loss: 0.3605 - learning_r
ate: 1.0000e-05
Epoch 46/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 171ms/step - accuracy: 0.9547 - loss: 0.1974
Epoch 46: val_accuracy did not improve from 0.89333
240/240 ━━━━━━━━━━━━━━━━━━━━ 81s 203ms/step - accuracy: 0.9547 - loss: 0.1975 - val_accuracy: 0.8926 - val_loss: 0.3604 - learning_r
ate: 1.0000e-05
Epoch 47/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 169ms/step - accuracy: 0.9539 - loss: 0.1988
Epoch 47: val_accuracy did not improve from 0.89333
240/240 ━━━━━━━━━━━━━━━━━━━━ 51s 212ms/step - accuracy: 0.9539 - loss: 0.1989 - val_accuracy: 0.8933 - val_loss: 0.3602 - learning_r
ate: 1.0000e-05
Epoch 48/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 173ms/step - accuracy: 0.9543 - loss: 0.1977
Epoch 48: val_accuracy did not improve from 0.89333
240/240 ━━━━━━━━━━━━━━━━━━━━ 52s 215ms/step - accuracy: 0.9543 - loss: 0.1977 - val_accuracy: 0.8933 - val_loss: 0.3603 - learning_r
ate: 1.0000e-05
Epoch 49/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 171ms/step - accuracy: 0.9546 - loss: 0.1973
Epoch 49: val_accuracy did not improve from 0.89333
240/240 ━━━━━━━━━━━━━━━━━━━━ 81s 213ms/step - accuracy: 0.9546 - loss: 0.1973 - val_accuracy: 0.8933 - val_loss: 0.3601 - learning_r
ate: 1.0000e-05
Epoch 50/50
239/240 ━━━━━━━━━━━━━━━━━━━━ 0s 165ms/step - accuracy: 0.9544 - loss: 0.1969
Epoch 50: val_accuracy did not improve from 0.89333
```

```
240/240 ━━━━━━━━━━━━━━━━━━━━ 81s 208ms/step - accuracy: 0.9544 - loss: 0.1969 - val_accuracy: 0.8926 - val_loss: 0.3600 - learning_r
ate: 1.0000e-05
```

In [50]:

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(EPOCHS)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



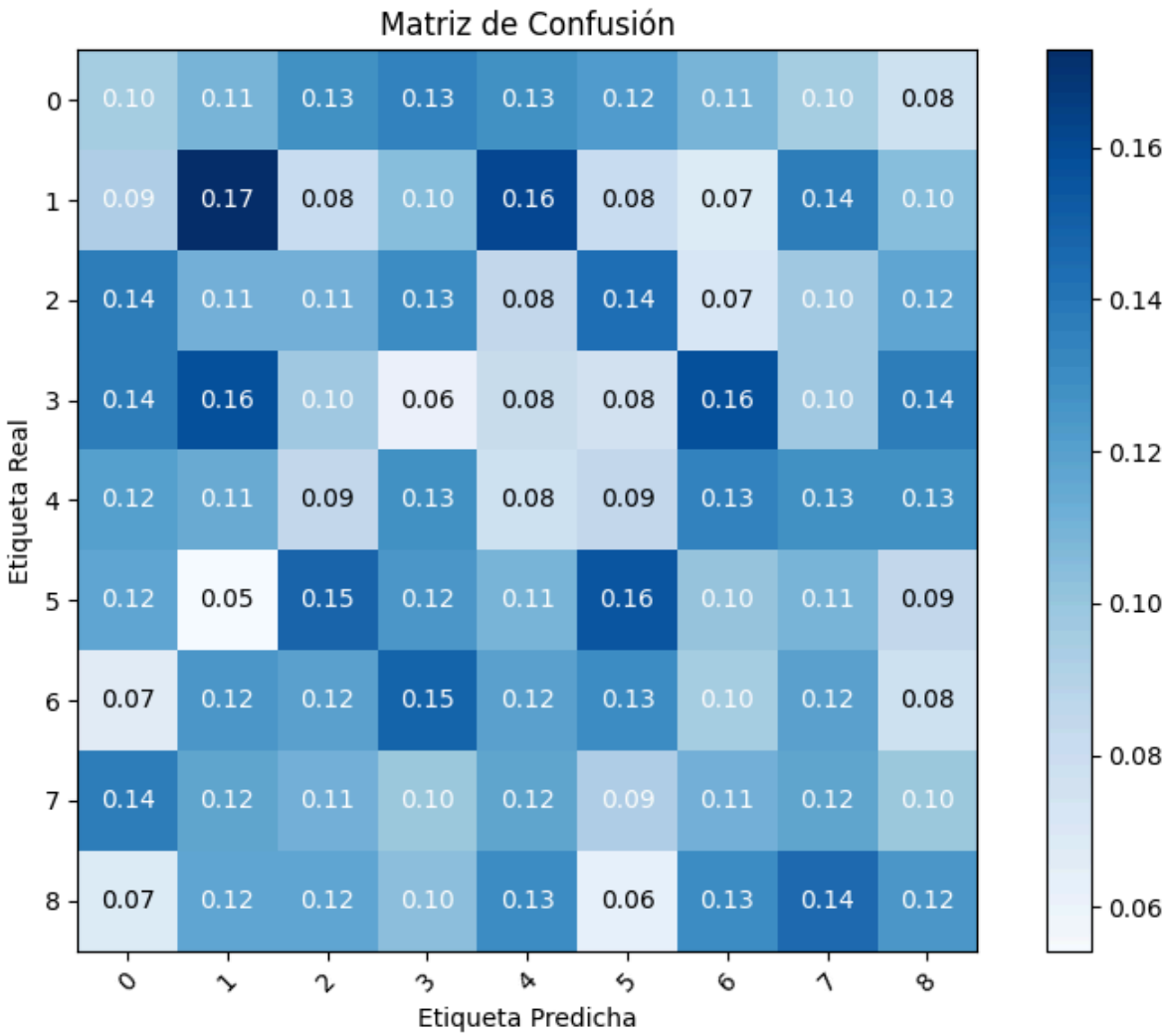
In [56]:
```
# Evaluar el modelo y obtener predicciones en el conjunto de prueba
y_true = np.concatenate([y for x, y in dataset_testing], axis=0)  # Etiquetas reales del conjunto de prueba
y_pred = model.predict(dataset_testing)  # Predicciones del modelo
y_pred = np.argmax(y_pred, axis=1)  # Convierte las probabilidades en etiquetas si es softmax
```

```
label_to_index = {label: idx for idx, label in enumerate(class_names)}
labels = list(label_to_index.values())  # Esto toma solo los valores numéricos del diccionario
labels
```

```
43/43 ━━━━━━━━━━━━━━━━━━━━ 10s 232ms/step
```

Out[56]: [0, 1, 2, 3, 4, 5, 6, 7, 8]

In [57]:
```
# Mostrar la matriz de confusión
mostrar_matriz_confusion(y_pred, y_true, labels=labels)  # Cambia los labels según tus clases
```



Resultado :

- El modelo tiene un buen desempeño, pero muestra señales leves de sobreajuste. La diferencia entre la precisión en entrenamiento y en validación, junto con la estabilización de la pérdida de validación, indica que el modelo podría estar aprendiendo detalles específicos del conjunto de entrenamiento que no son tan útiles para datos nuevos.

Evaluación.

In [53]:
```
# Crear un archivo para guardar los pesos del modelo
top_model_weights_path = '/content/drive/My Drive/IA/skin_cancer_model.weights.h5'
model.save_weights(top_model_weights_path)

# Crear un archivo para guardar el modelo completo
top_model_path = '/content/drive/My Drive/IA/skin_cancer_model.h5'
model.save(top_model_path)
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is c
onsidered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_mode
l(model, 'my_model.keras')`.
```

In [54]:
```
(eval_loss, eval_accuracy) = model.evaluate(dataset_testing, batch_size=batch_size, verbose=1)
```

```
43/43 ━━━━━━━━━━━━━━━━━━━━ 8s 189ms/step - accuracy: 0.8950 - loss: 0.3667
```

In [55]:
```
print("[INFO] accuracy: {:.2f}%".format(eval_accuracy * 100))
print("[INFO] Loss: {}".format(eval_loss))
```

```
[INFO] accuracy: 89.26%
[INFO] Loss: 0.360038161277771
```

In [ ]: