In [1]:

```
# TensorFlow y tf.keras
import tensorflow as tf
from tensorflow import keras

# Librerias de ayuda
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)
```

```
2.9.2
```

In [2]:

```
fashion_mnist = keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-l
abels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-i
mages-idx3-ubyte.gz
26421880/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-la
bels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-im
ages-idx3-ubyte.gz
4422102/4422102 [==============================] - 0s 0us/step
```

In [3]:

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

In [4]:

```
train_images.shape
```

Out[4]:

```
(60000, 28, 28)
```

In [7]:

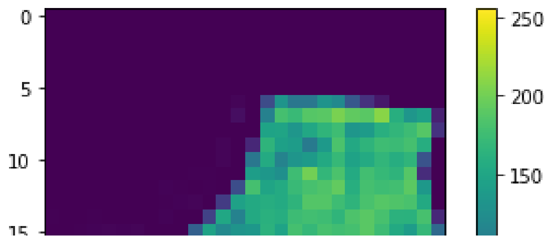
```
test_images.shape
```

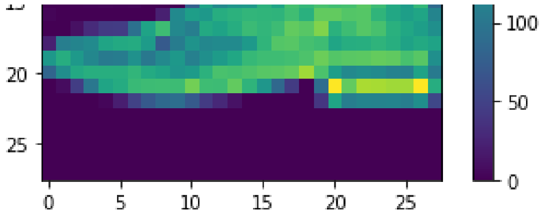
Out[7]:

```
(10000, 28, 28)
```

In [12]:

```
plt.figure()
plt.imshow(train_images[1700])
plt.colorbar()
plt.grid(False)
plt.show()
```



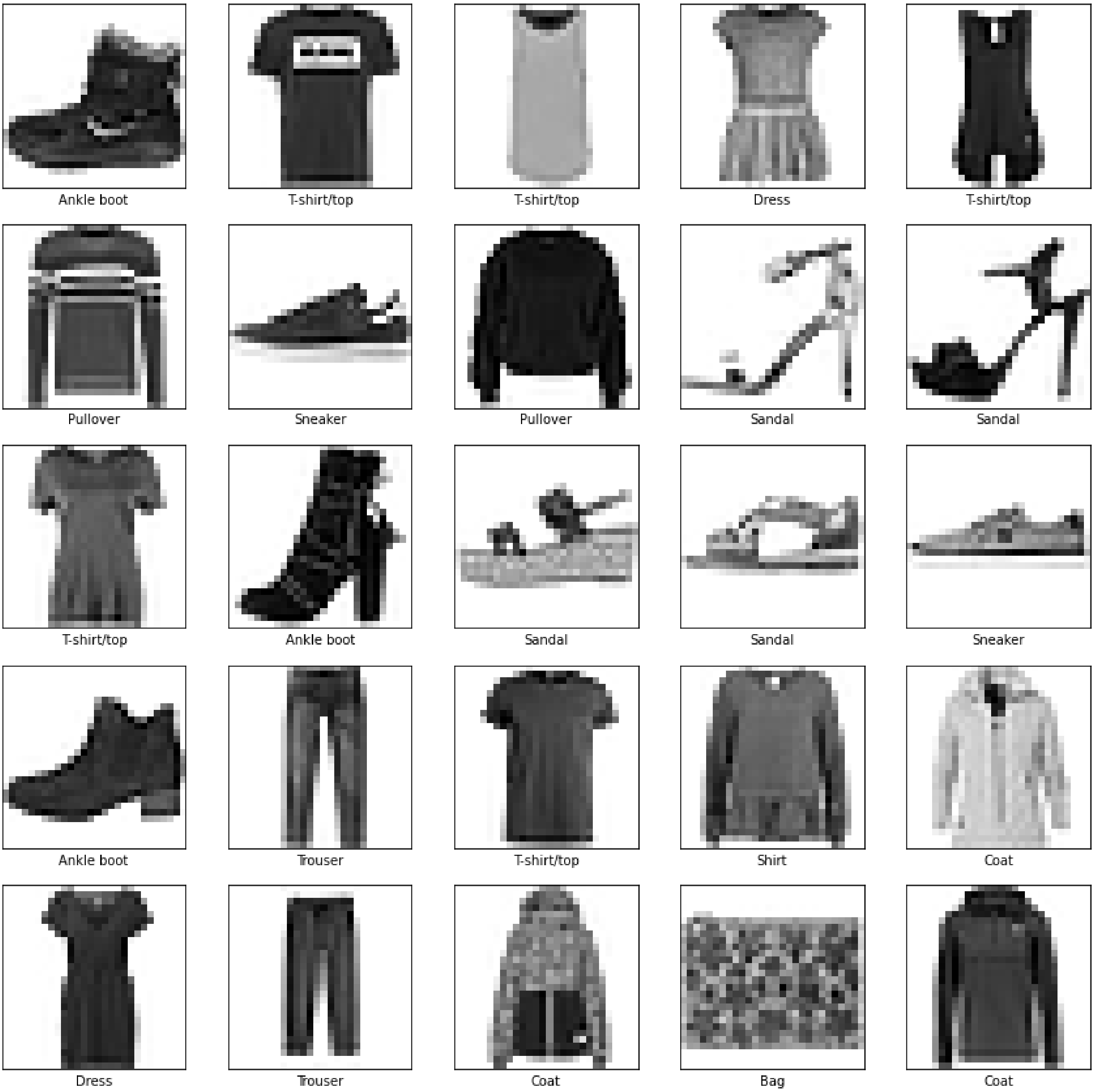


In [16]:

```
train_images = train_images / 255.0

test_images = test_images / 255.0
```

In [21]:

```
plt.figure(figsize=(15,15))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```



In [22]:

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])
```

In [23]:

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

In [24]:

```
model.fit(train_images, train_labels, epochs=20)
```

```
Epoch 1/20
1875/1875 [==============================] - 12s 6ms/step - loss: 0.5011 - accuracy: 0.82
43
Epoch 2/20
1875/1875 [==============================] - 10s 5ms/step - loss: 0.3728 - accuracy: 0.86
53
Epoch 3/20
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3357 - accuracy: 0.877
7
Epoch 4/20
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3120 - accuracy: 0.885
5
Epoch 5/20
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2916 - accuracy: 0.891
9
Epoch 6/20
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2793 - accuracy: 0.896
7
Epoch 7/20
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2662 - accuracy: 0.900
9
Epoch 8/20
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2544 - accuracy: 0.904
9
Epoch 9/20
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2444 - accuracy: 0.908
8
Epoch 10/20
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2370 - accuracy: 0.912
6
Epoch 11/20
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2290 - accuracy: 0.915
5
Epoch 12/20
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2204 - accuracy: 0.917
0
Epoch 13/20
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2145 - accuracy: 0.919
2
Epoch 14/20
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2089 - accuracy: 0.922
0
Epoch 15/20
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2013 - accuracy: 0.924
4
Epoch 16/20
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1960 - accuracy: 0.926
3
Epoch 17/20
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1889 - accuracy: 0.929
5
Epoch 18/20
1875/1875 [==============================] - 8s 4ms/step - loss: 0.1861 - accuracy: 0.930
```

```
1875/1875 [==============================] - 8s 4ms/step - loss: 0.1861 - accuracy: 0.930
7
Epoch 19/20
1875/1875 [==============================] - 7s 3ms/step - loss: 0.1804 - accuracy: 0.932
8
Epoch 20/20
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1736 - accuracy: 0.934
7
```

Out[24]:

```
<keras.callbacks.History at 0x7f9621ebcc50>
```

In [25]:

```
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

print('\nTest accuracy:', test_acc)
```

```
313/313 - 1s - loss: 0.3592 - accuracy: 0.8833 - 774ms/epoch - 2ms/step

Test accuracy: 0.8833000063896179
```

In [26]:

```
predictions = model.predict(test_images)
```

```
313/313 [==============================] - 2s 4ms/step
```

In [27]:

```
predictions[0]
```

Out[27]:

```
array([2.2701953e-11, 5.3148228e-18, 8.1085675e-16, 9.2548321e-13,
       5.6857997e-15, 3.2855562e-05, 4.1287340e-10, 5.1345192e-03,
       1.7807861e-14, 9.9483258e-01], dtype=float32)
```

In [28]:

```
np.argmax(predictions[0])
```

Out[28]:

```
9
```

In [29]:

```
test_labels[0]
```

Out[29]:

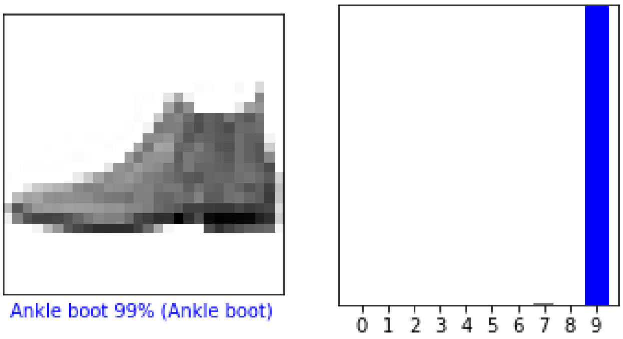
```
9
```

In [30]:

```
def plot_image(i, predictions_array, true_label, img):
  predictions_array, true_label, img = predictions_array, true_label[i], img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

  plt.imshow(img, cmap=plt.cm.binary)

  predicted_label = np.argmax(predictions_array)
  if predicted_label == true_label:
    color = 'blue'
  else:
    color = 'red'

  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                100*np.max(predictions_array),
                                class_names[true_label]),
                                color=color)

def plot_value_array(i, predictions_array, true_label):
  predictions_array, true_label = predictions_array, true_label[i]
  plt.grid(False)
  plt.xticks(range(10))
  plt.yticks([])
  thisplot = plt.bar(range(10), predictions_array, color="#777777")
  plt.ylim([0, 1])
  predicted_label = np.argmax(predictions_array)

  thisplot[predicted_label].set_color('red')
  thisplot[true_label].set_color('blue')
```

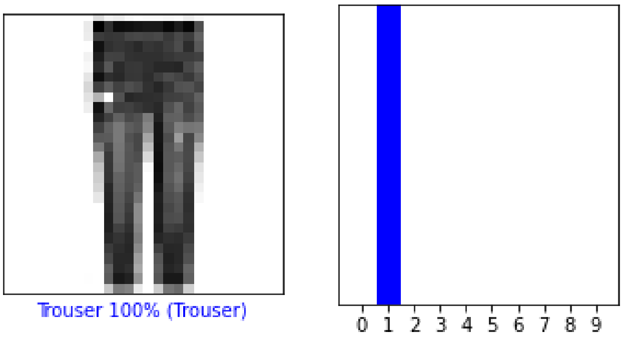
In [31]:

```
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()
```



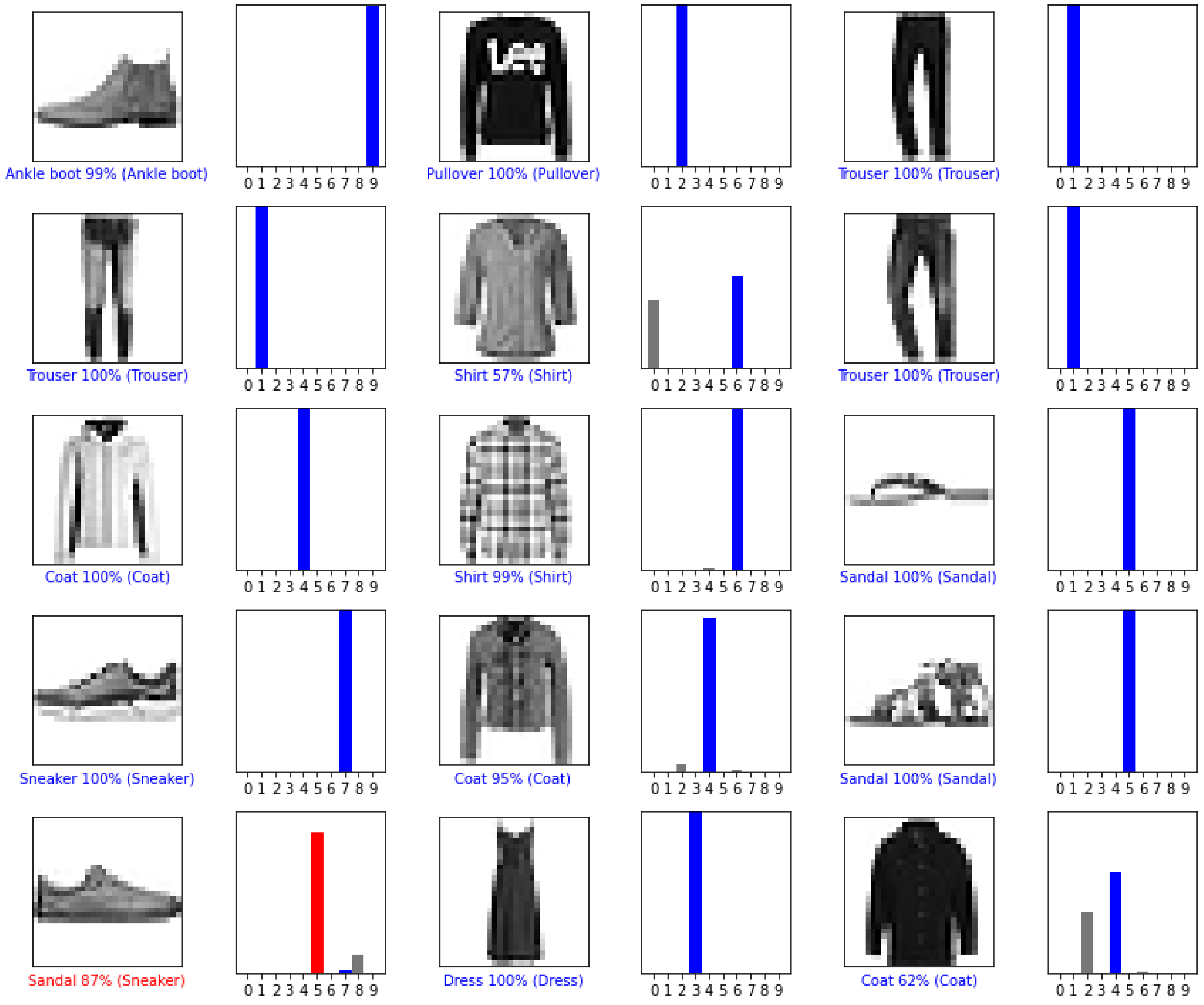
In [40]:

```
i = 200
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()
```



In [33]:

```
# Plot the first X test images, their predicted labels, and the true labels.
# Color correct predictions in blue and incorrect predictions in red.
num_rows = 5
num_cols = 3
num_images = num_rows*num_cols
plt.figure(figsize=(2*2*num_cols, 2*num_rows))
for i in range(num_images):
  plt.subplot(num_rows, 2*num_cols, 2*i+1)
  plot_image(i, predictions[i], test_labels, test_images)
  plt.subplot(num_rows, 2*num_cols, 2*i+2)
  plot_value_array(i, predictions[i], test_labels)
plt.tight_layout()
plt.show()
```


Ankle boot 99% (Ankle boot)
Pullover 100% (Pullover)
Trouser 100% (Trouser)
Trouser 100% (Trouser)
Shirt 57% (Shirt)
Trouser 100% (Trouser)
Coat 100% (Coat)
Shirt 99% (Shirt)
Sandal 100% (Sandal)
Sneaker 100% (Sneaker)
Coat 95% (Coat)
Sandal 100% (Sandal)
Sandal 87% (Sneaker)
Dress 100% (Dress)
Coat 62% (Coat)