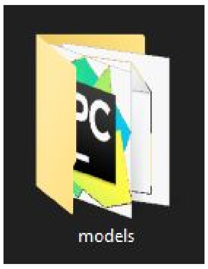
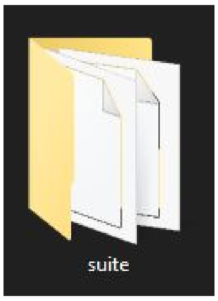
## Project libraries structure

**The root folder (TestAutomation) contains:**



- **models**

In this folder is where we'd get the files that are going to help us with the repetitive tasks, it's accessed for all the classes to gather the methods, it's my principal library where all the classes to get the necessary functions. It has to be made before all the programming because here we'd detect the main processes and therefore find these classes and methods that could be called many times and reuse them or just to have them separate and call them only when it's necessary.



- **suite**

Here is where we'd get all the scripts that we're going to run, these scripts are based on test cases and requirements but the classes are independent each other, maybe they could use similar functions from the same library but they don't share data between them, it contains all the methods necessary to run in correct form the script.

- **test**

This folder is necessary to run all the scripts, it contains the necessary files to get an automated (using pytest) or manual testing and show the results, it's our main menu and gather all the models and suite folder classes stuff that are necessary to run our suite.

With the **model** folder help, we get all the methods, in **suite** folder is where we have the suite information and the **test** folder, finally, let us run all of them and get the results.

## Project methods

**suite_info:**
Suite information (name and version)
:return: append info # script: name version: version'

**suite_methods:**
suite methods that we're going to run

**get_time:**
the actual time when our script when we call it
:return: actual time

**test_local_call:**
Automated method using pytest for local calls and let us to run it without input by the user, it's introduced by this library.
:return: the result of the suite

**test_national_call:**
Automated method using pytest for national calls and let us to run it without input by the user, it's introduced by this library.
:return: the result of the suite

**suite_execution:**
Gather all the scripts, run them and show the results

**read_serial:**
Method that reads the serial of the first android device detected by adb
on the send position ::default = 1
:return: serial of the detected device

**get_serial_and_device:**
Method that get and return the serial and device
:return: serial and device

**open_app:**
Method that retrieves all the processes to open menu
:return: when the process ends

**validate_number:**
Method that validates the length and regex of the number (only: digits and characters)
:return: validation of the number entered

**call_adb_number:**
Method that retrieves the methods that allow us to do the call and run them in the correct order to make the call
and append the number to commit the adb command

**get_number:**
Method that retrieves the user input of the number to dial
:return: a string variable of the dial number

**clear_phone_display:**
Method that help us in phone display and delete all the input numbers over there, starting with a sample number
to get the option to backspace

**call_number:**
Method that retrieves all the processes to ask a number and call it
:return: when the process ends

**toggle_wifi_status:**
Method that retrieves all the processes to toggle our wifi status
:return: when the function is completed

**get_wifi_status: # 0 == turn wifi off | 1 == turn wifi on**
Method that checks if the wifi is on or off and decides through what returns, if it changes the state or leaves it as it is
:return: decision based on wifi status

**wait:**
Method used default waiting time = "0.01"

**wait_process_completed:**
Method that wait until process is completed and go back to menu