Framework UI Automator  
Documentación

22 de mayo de 2020

# Documentación

## Arquitectura de Software

### Ui Automator

Python Scripts

UI Automator

Android Cellphone

### Twilio

Python Scripts

UI Automator

Android Cellphone

Twilio

## Explicación Código

### adb\_utils.py

Incluye distintos métodos que pueden ser útiles para diferentes scripts usando las funciones de ADB, incluye los siguientes métodos:

* **read\_device**(int j) -> string: Regresa el serial del j-esimo dispositivo conectado al pc.
* **read\_devices**() -> list<string>: Regresa los serials de todos los dispositivos conectados.
* **adb\_call**(phone\_number: string, seconds: float, serial: string): Realiza una llamada al número *phone\_number* con una duración de *seconds* segundos usando comando de adb shell.

### constants.py

Incluye diferentes diccionarios que son utilizados por otras librerías.

### credentials.py

Incluye las credenciales para twilio.

### ui\_utils.py

Incluye distintos métodos que pueden ser útiles para diferentes scripts usando las funciones de ADB y uiautomator, incluye los siguientes métodos:

* **adb\_ui\_call**(phone\_number: string, seconds: float, device: uiautomator.Device, serial: string): Simula el proceso de llamada a *phone\_number* por *seconds* segundos a través de la interface del celular.
* **adb\_ui\_wifi**(status: {0,1}, device: uiautomator.Device, serial: string) Simula el proceso de encendido o apagado de wifi a través de la interface del celular según el valor de status, 1 para encender y 0 para apagar, además se regresa un mensaje en caso de que se quiera cambiar el estatus de wifi así mismo.
* **adb\_ui\_calculator**(operand1: number, operator:{-,+,\*,/}, operand2:number, device: uiautomator.device): Simula una operación dada los operandos y los operadores, soporta números negativos y decimales.
* **adb\_ui\_voice\_message**(device: uiautomator.device): Verifica si el celular tiene una notificación de mensaje de voz.

### twilio\_utils.py

Incluye métodos usando el api de twilio:

* **make\_call**(message: string, \_to: number, \_from: number): Genera un mensaje de voz desde \_from hacia \_to con el audio de message, el receptor no debe contestar la llamada.

### logger.py

* **create\_file**(file\_name: string, headers: list<string>): Crea un archivo file\_name en la carpeta log/.
* **write\_file**(file\_name: string, headers: list<string>, row: string): Agrega row al final del arvhico file\_name.

#### Nota

Las funciones de ui\_utils y adb\_utils incluyen parámetros opcionales para testing:

* **debug**: Booleano, en caso de que se quiera hacer testing el valor debe ser true
* **log\_file:** El archivo donde se guardará la información de la prueba.

### Scripts

* **adb\_call:** Ejemplo de uso de adb\_call.
* **adb\_ui\_call:** Ejemplo de uso de adb\_ui\_call.
* **adb\_ui\_wifi\_off:** Ejemplo de uso de adb\_ui\_wifi para apagar el wifi.
* **adb\_ui\_wifi\_on:** Ejemplo de uso de adb\_ui\_wifi para encender el wifi.
* **data:** Incluye una lista con toda la información de las pruebas a probar.
* **main:** Ejecuta para caso la función a probar con sus parámetros correspondientes
* **log/:** En dicha carpeta se guardan los archivos con los resultados de las pruebas probados

## Creación de caso prueba.

Se deberá agregar un diccionario el siguiente a la lista en data.py

{  
 'function': 'adb\_ui\_wifi',  
 'description': 'Apagar wifi con wifi encendido',  
 'automated': 'True',  
 'parameters': {  
 'device\_id': '1',  
 'status': '0',  
 },  
 'expected result': 'Estatus de Wifi cambiado a Off'  
},

* **function:** Los valores para function pueden ser los siguientes:
  + read\_devices
  + read\_device
  + adb\_call
  + adb\_ui\_call
  + adb\_ui\_wifi
  + adb\_ui\_calculator
  + test\_adb\_ui\_voice\_message
* **description:** La descripción del caso prueba.
* **automated:** En caso de que se quiera ejecutar junto con la suite el valor deberá ser True, en caso contrario False.
* **parameters:** Revisar la documentación de cada función para ver el tipo de parámetros que recibe, no es necesario enviar serial o device, solo seleccionar el índice del teléfono donde se quiere ejecutar.
* **expected result:** La salida esperada al ejecutar la prueba.

## Setup

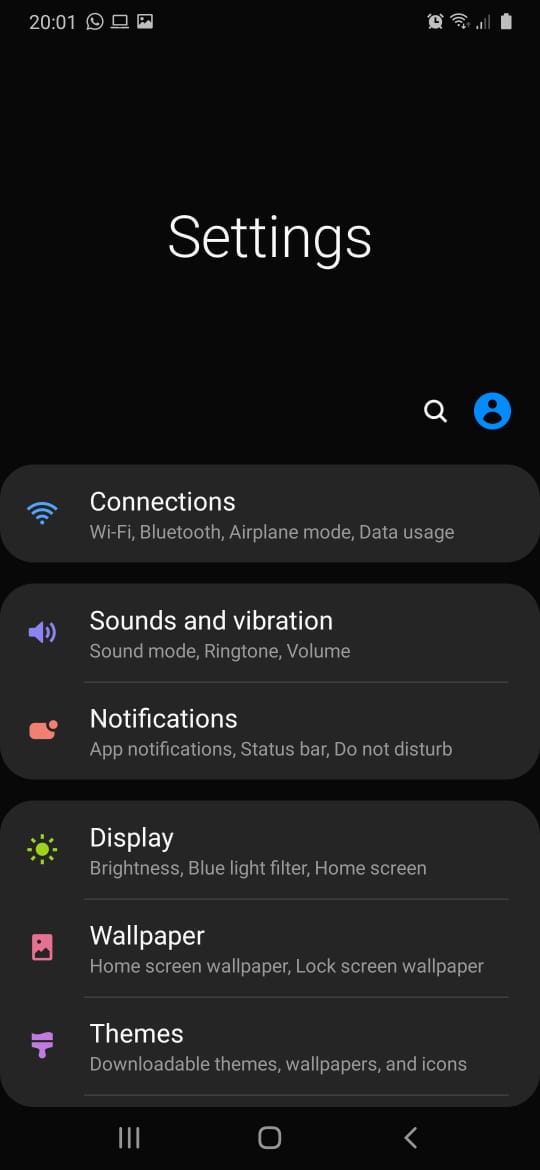
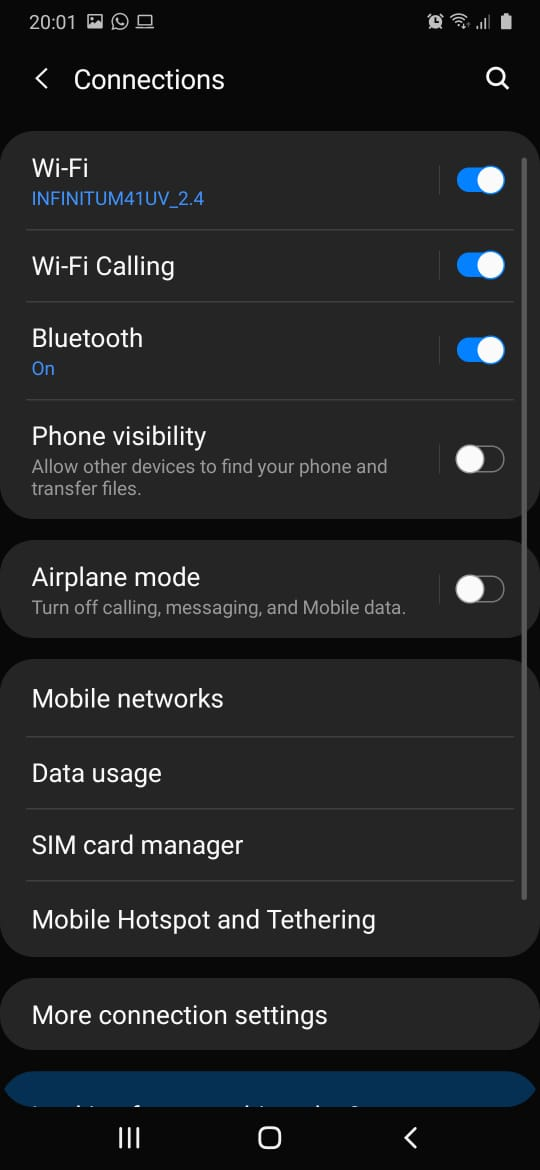
* [Instalar Python 2.7](https://www.python.org/downloads/release/python-2718/)
* [Instalar uiautomator en Python 2](https://github.com/xiaocong/uiautomator): pip install uiautomator
* [Instalar Android Studio con el paquete adb:](https://developer.android.com/studio/command-line/adb?hl=es-419) Desde el sdk manager de Android Studio seleccionar platform-tools.
* [Agregar adb como variable de entorno](https://lifehacker.com/the-easiest-way-to-install-androids-adb-and-fastboot-to-1586992378): Agregar C:\Program Files (x86)\Android\android-sdk\platform-tools a las variables de entorno desde la configuración avanzada del equipo.
* [Habilitar modo desarrollador en celular](https://developer.android.com/studio/debug/dev-options)
* [Instalar librería para Twilio](https://www.twilio.com/docs/libraries/python)
* [Agregar credentials.py](https://www.twilio.com/console)
* [Agregar número verificado para twilio](https://www.twilio.com/console/phone-numbers/verified)
* Colocar icono de Settings en la pantalla de home del celular.
* Colocar icono de Calculadora en la pantalla de home del celular.

## Ambiente de Dispositivo

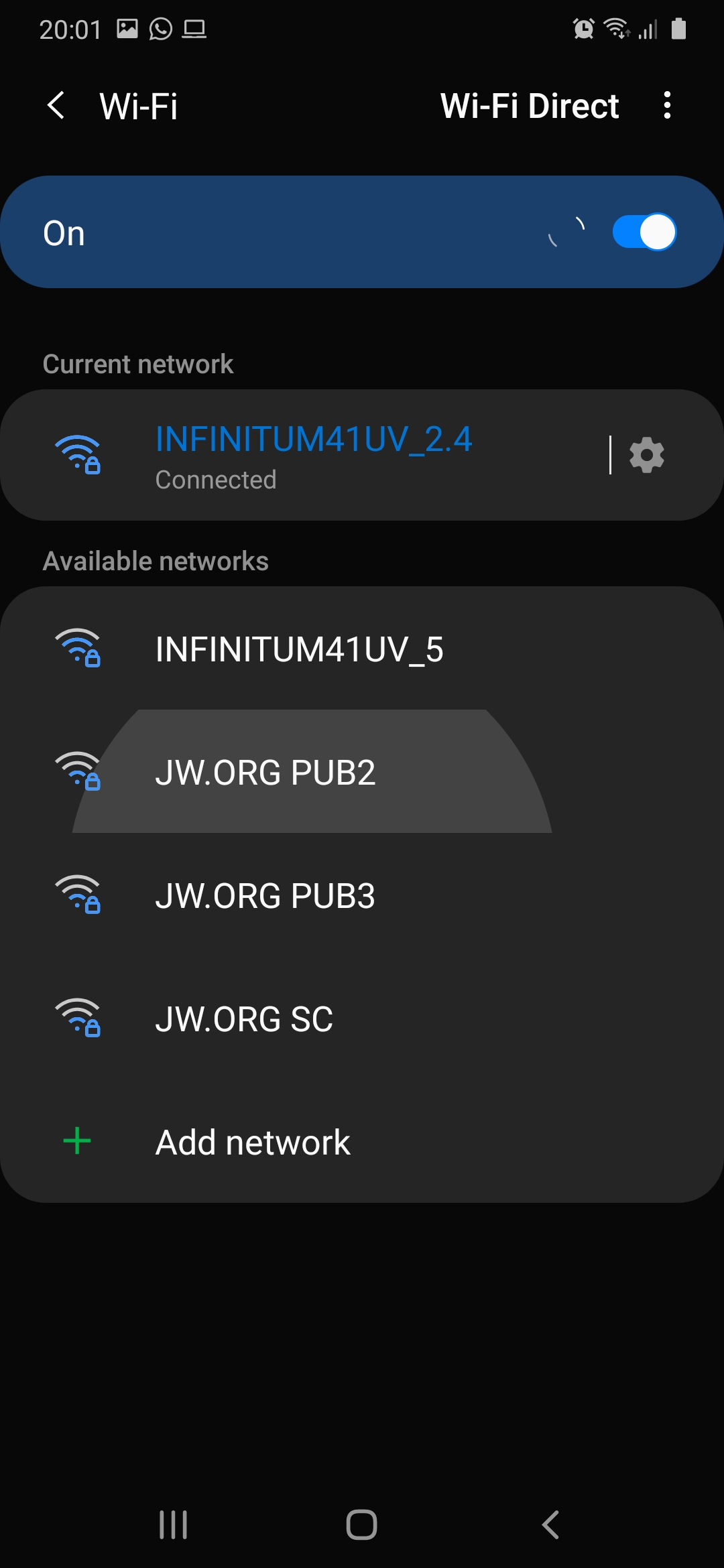
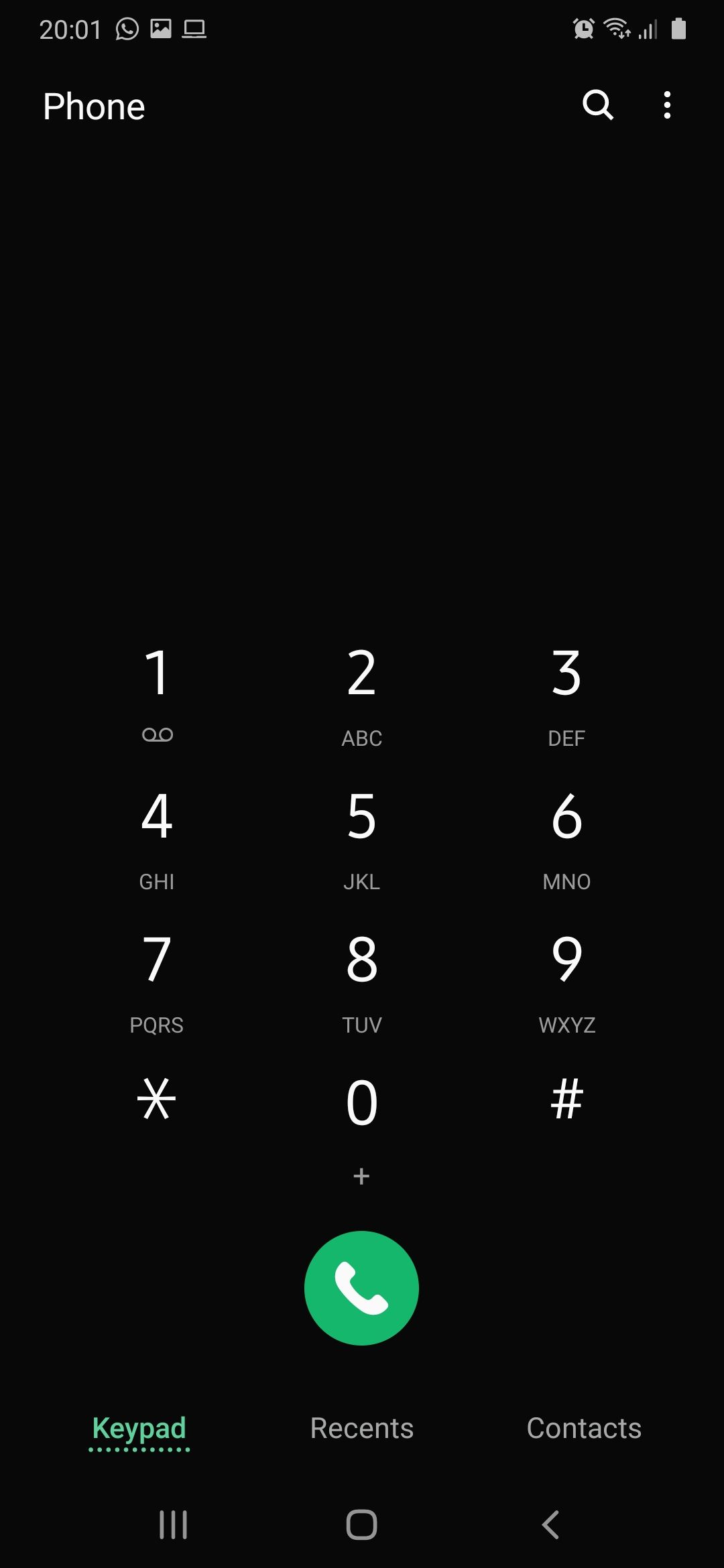
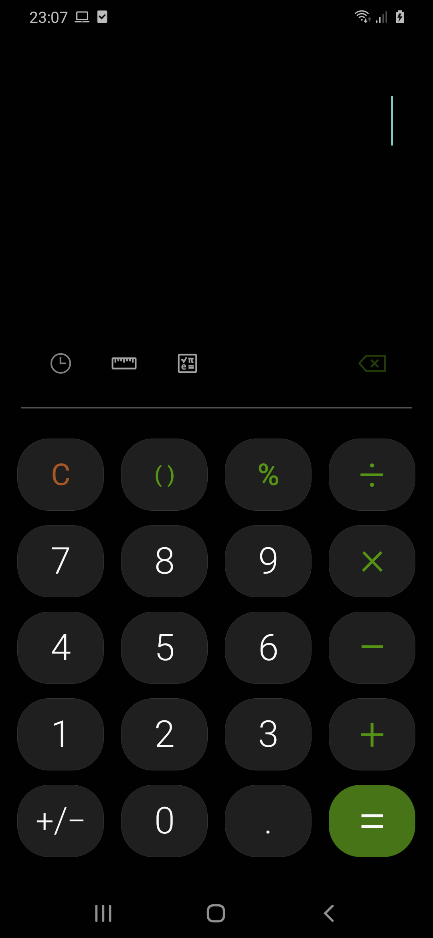
* **Modelo:** Galaxy A50 – SM-A505G
* **Serial:** R28M410479V
* **One UI version:** 2.0
* **Android version:** 10
* **Kernel versión:** 4.14.62-18098116 24/03/2020
* **Android security patch level:** 01/03/2020
* **Lenguaje:** Ingles

Pantallas

**Pantalla Home Pantalla Settings Pantalla Connections**

**Pantalla Wifi Pantalla Call/Keypad Pantalla Calculator**

**Pantalla Notifications Pantalla Notifications con Voicemail**

