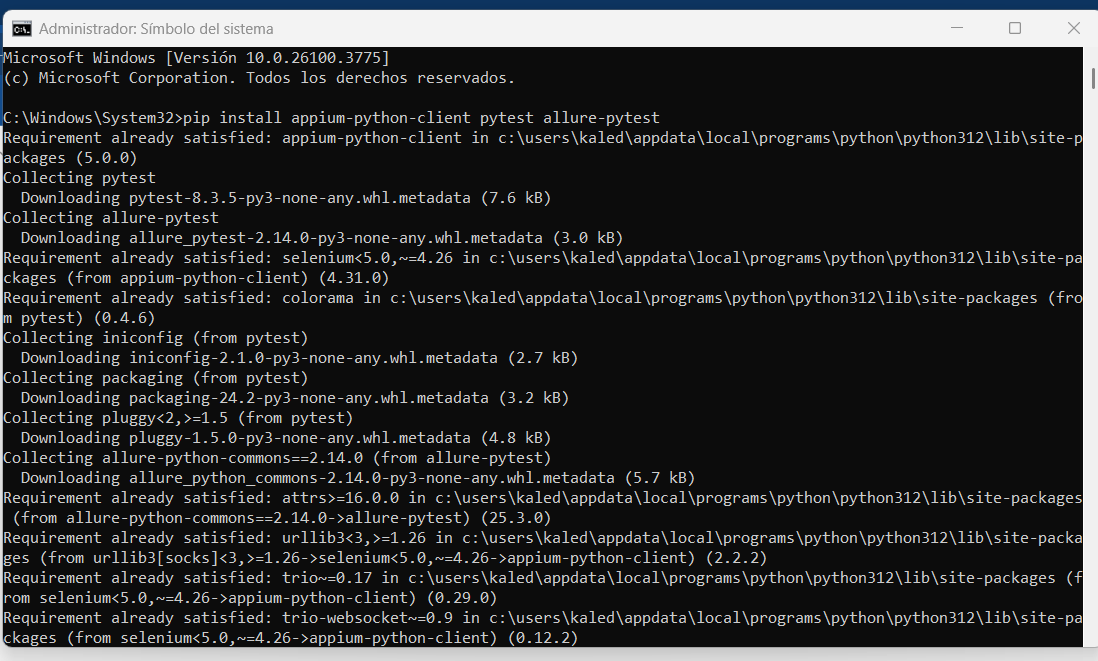
****

**stack de pruebas:**

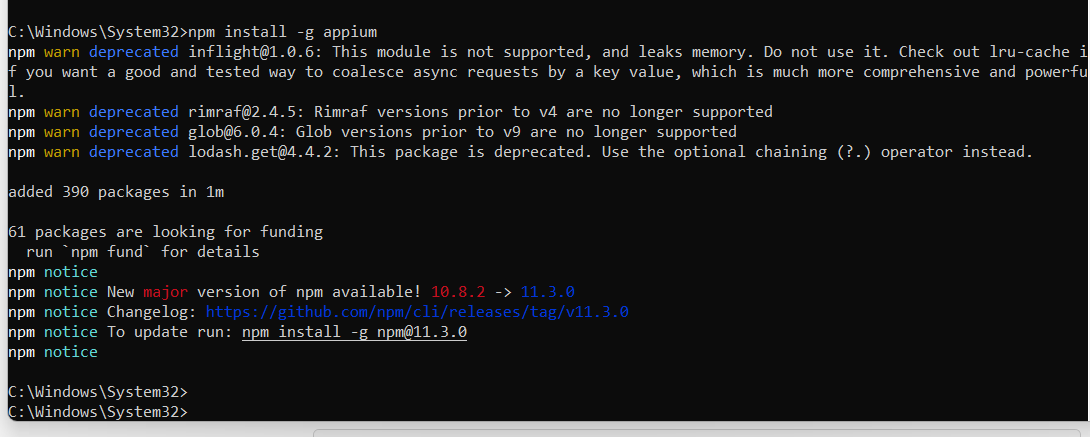
* **Python + Appium + Pytest** (o Unittest si prefieres).
* **Appium Server** ejecutándose localmente.
* El **APK** que quieres testear.
* Un **emulador Android** o un celular conectado por USB.

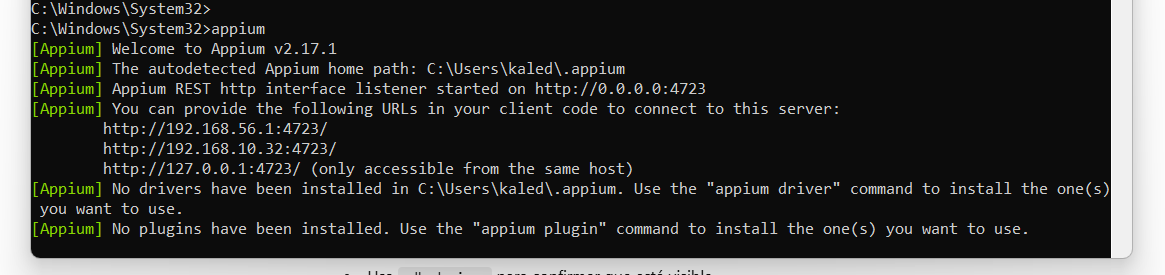
PASOS PARA EMPEZAR

pip install appium-python-client pytest allure-pytest

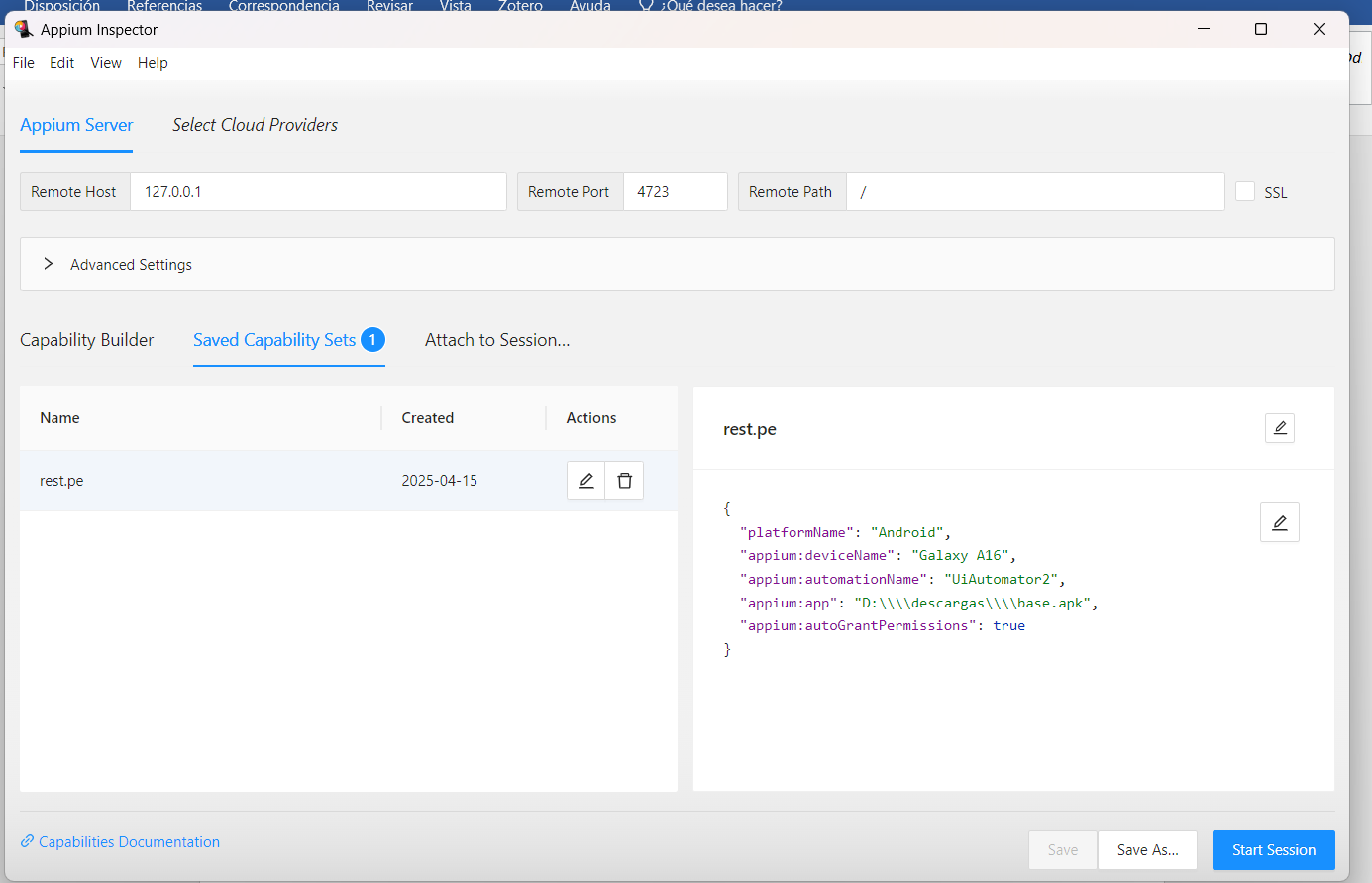


npm install -g appium

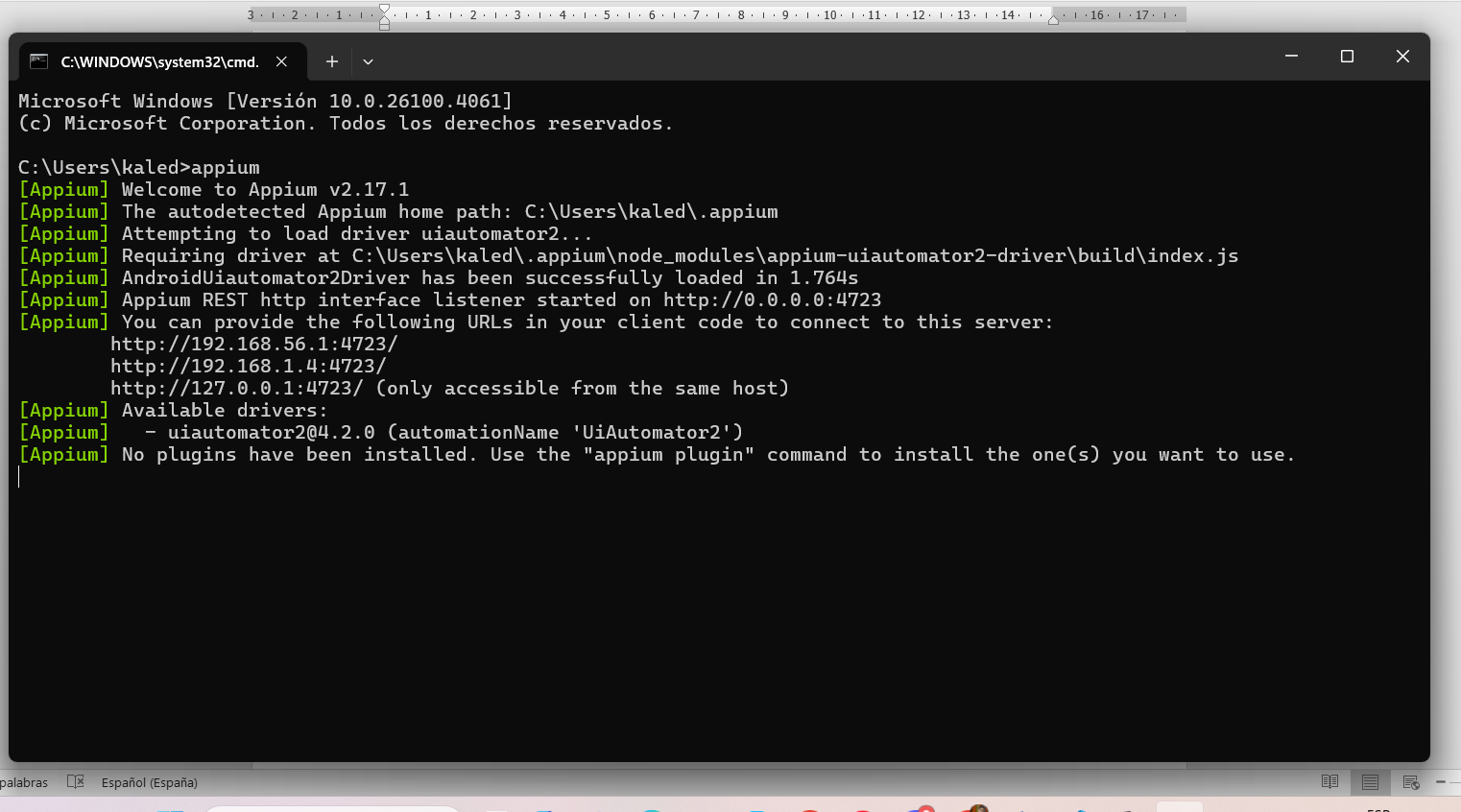


Luego: ejecutar APPIUM: 

Cargar la data en paquete Json:

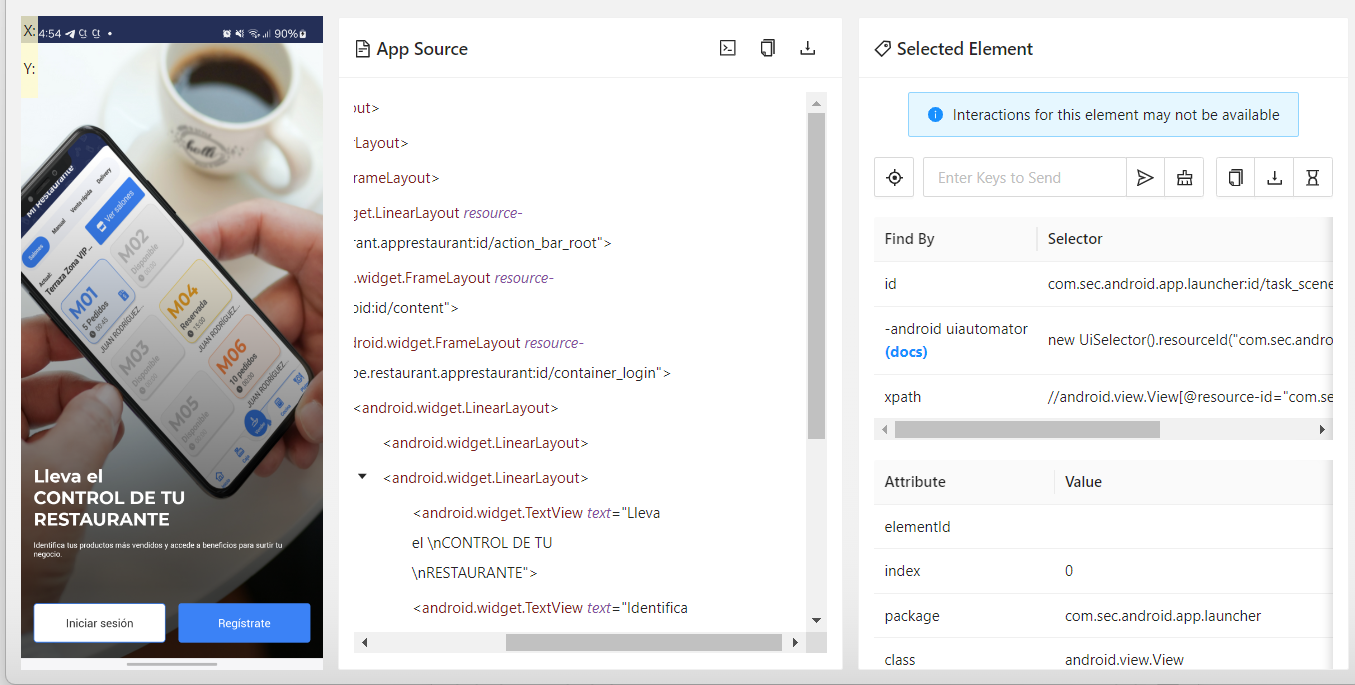


Levantamos el servidor en nuestra PC:



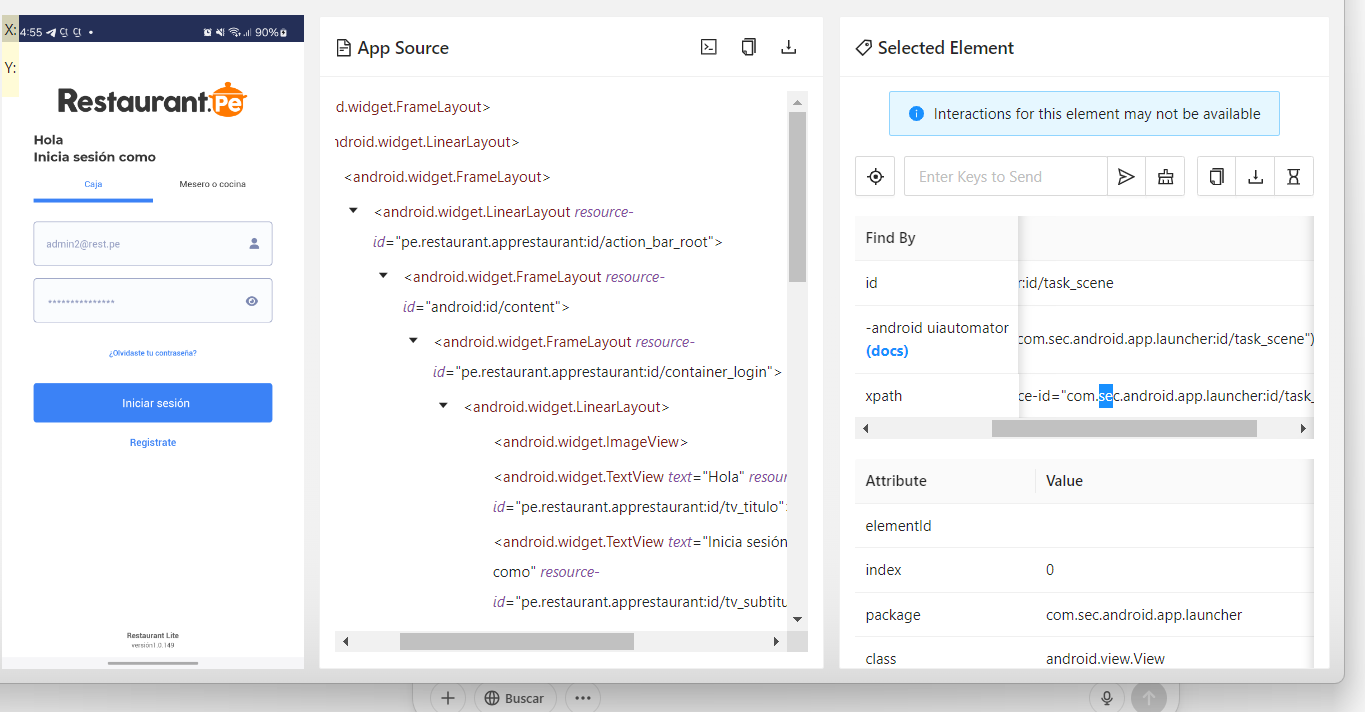
**BUSQUEDA DE CAUSISTICAS**

Al inicio hay un botón que **“Iniciar Sesión”**

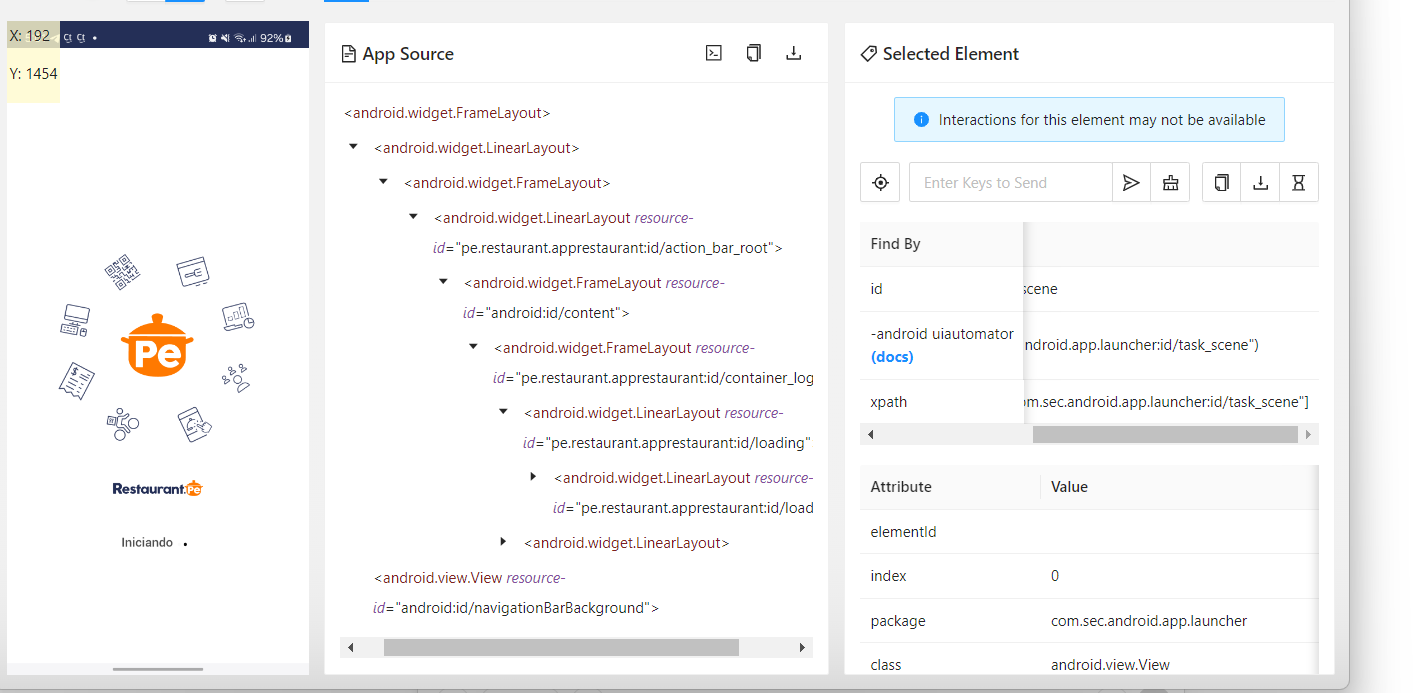


Luego nos muestra la forma en la que iniciamos sesión:

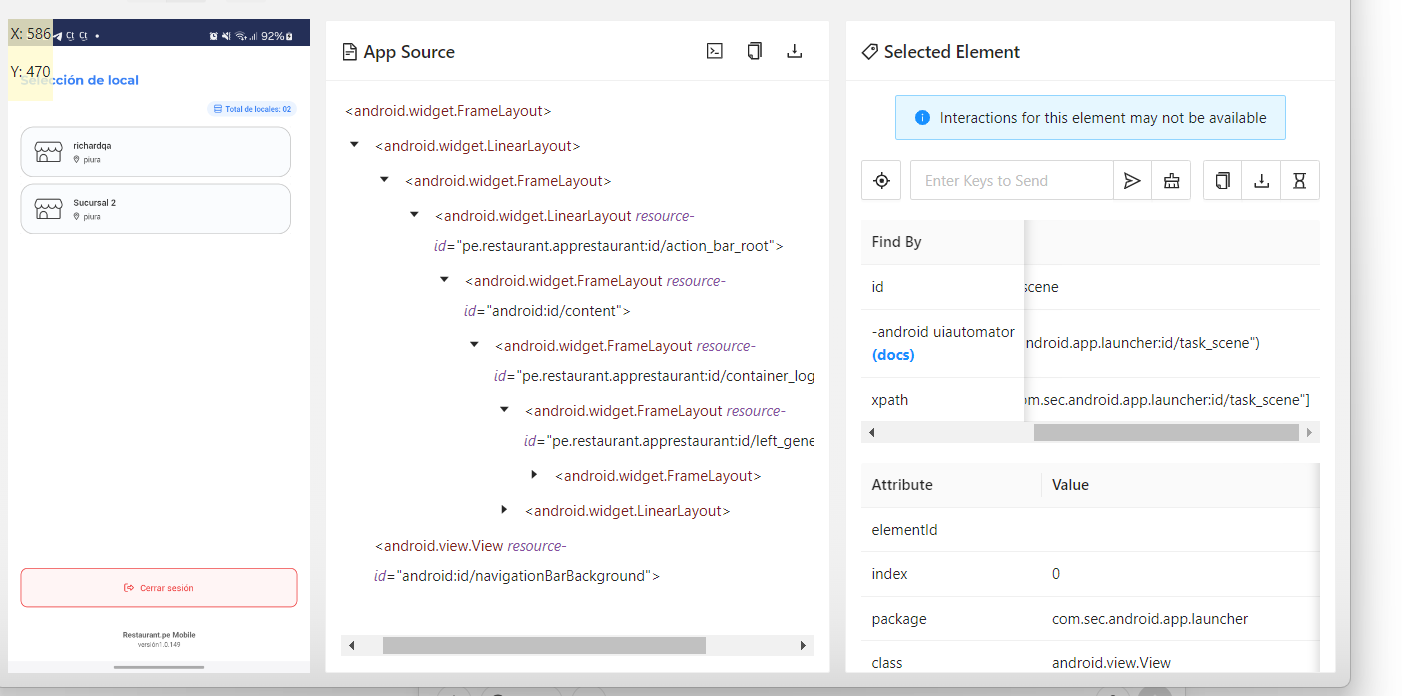
Ingresar datos, usuario y contraseña



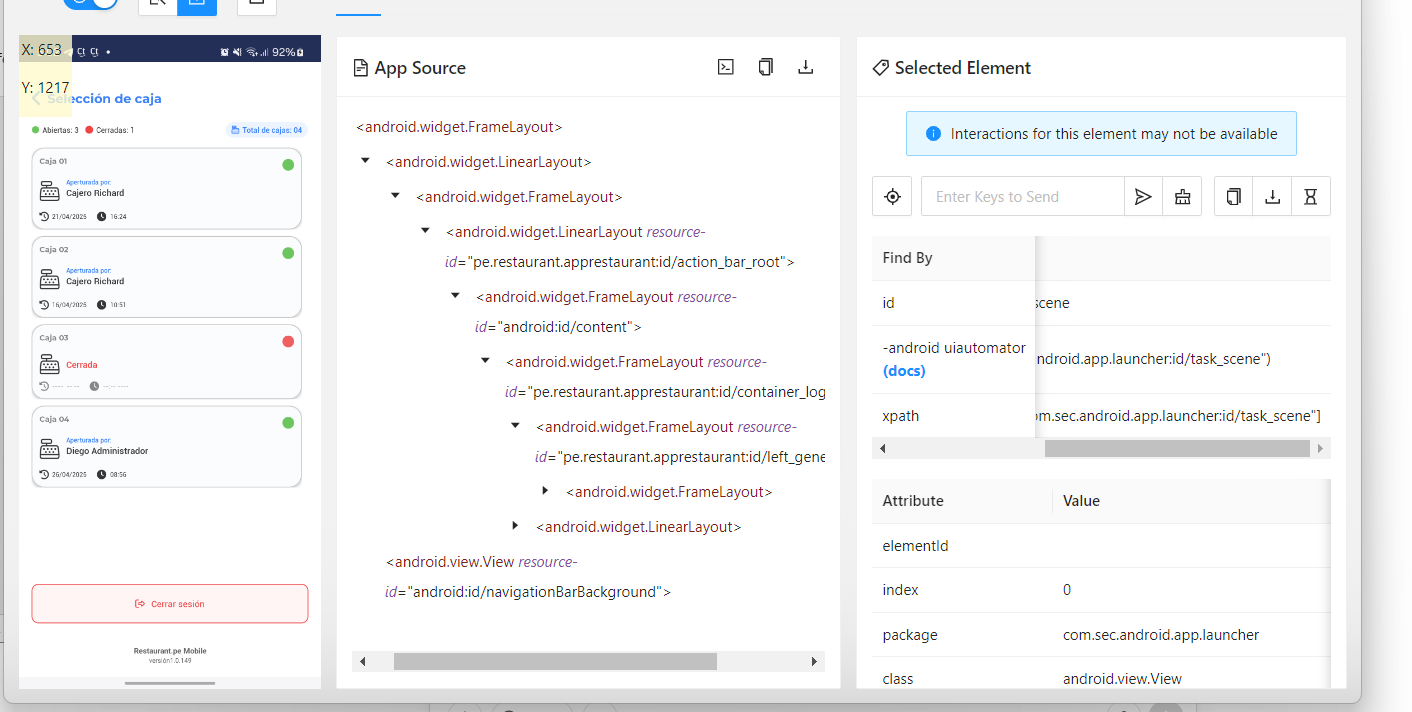
Luego carga esta imagen: permite sincronizar los datos:



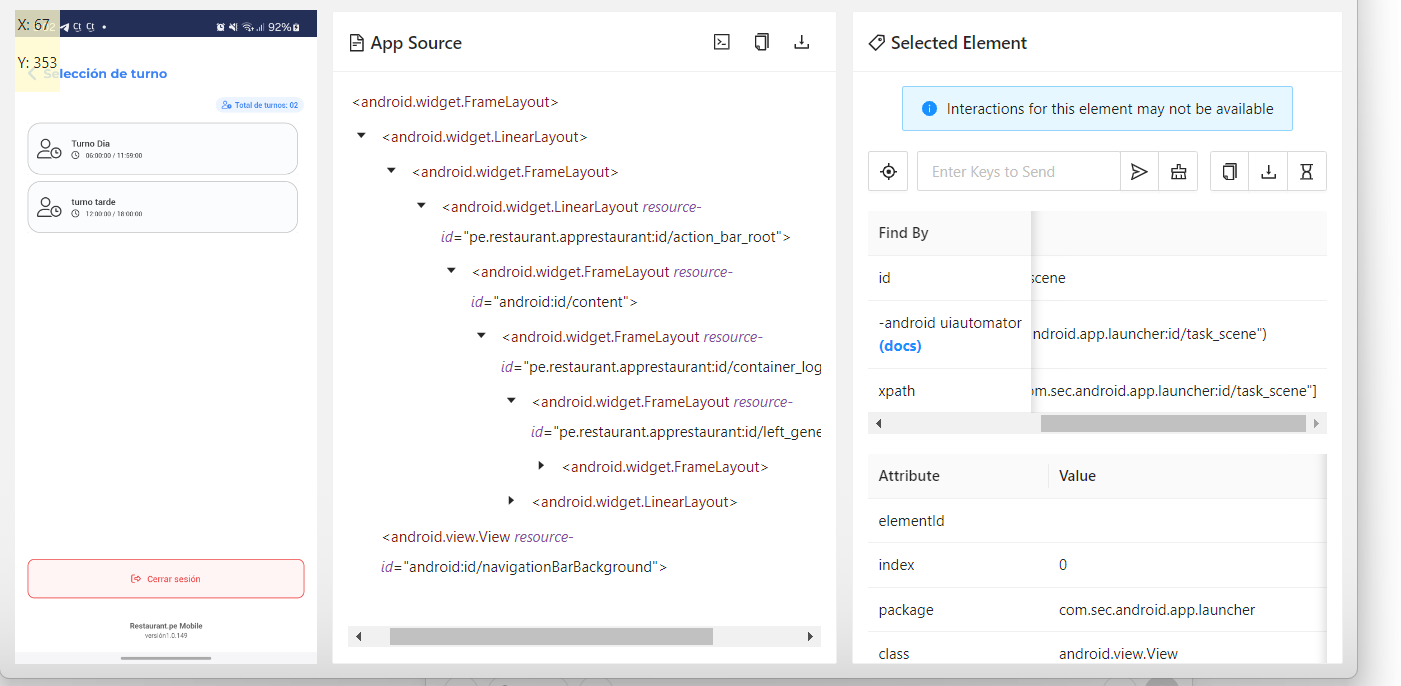
Luego se selecciona el local:



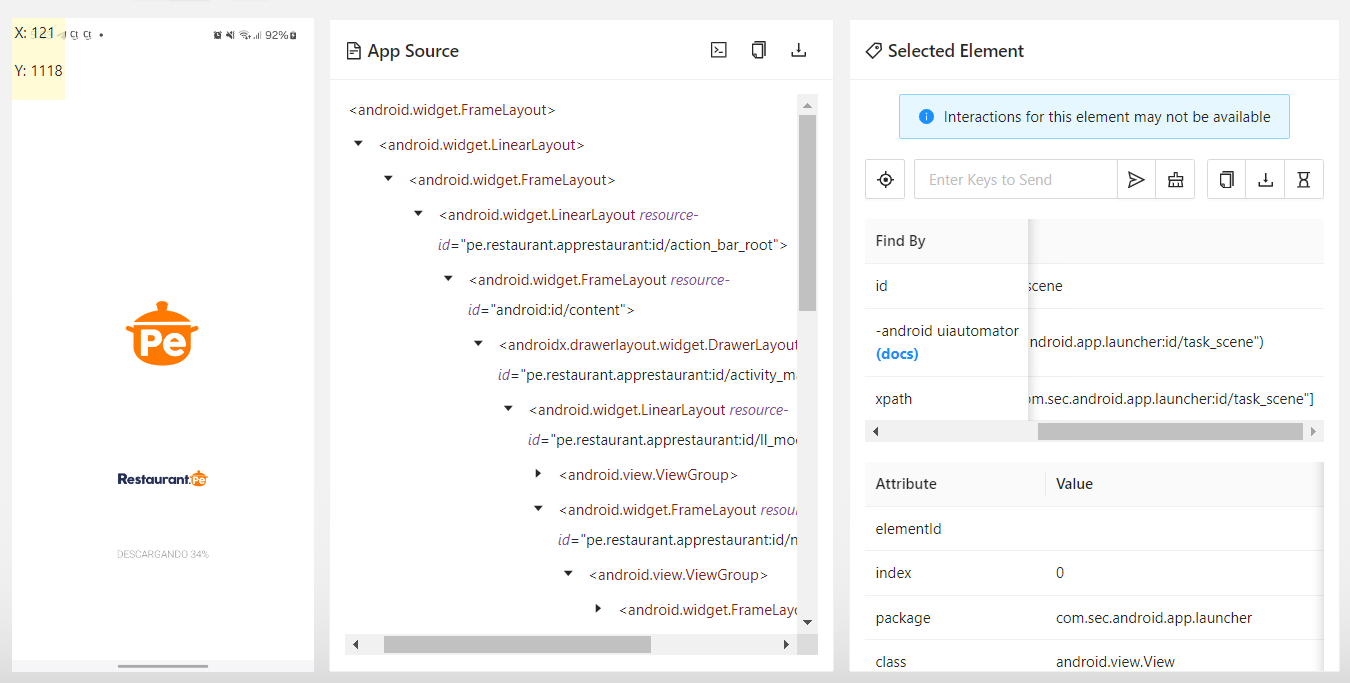
Luego seleccionamos una caja



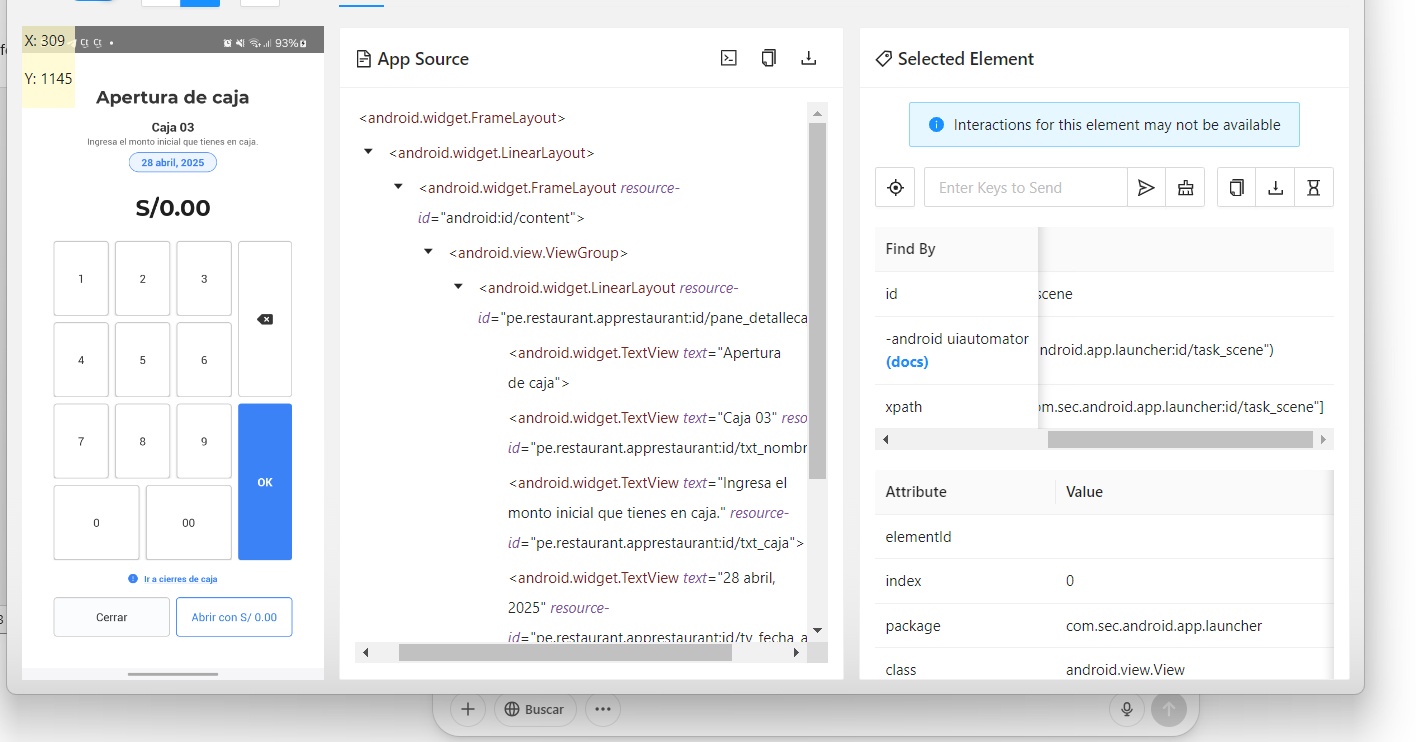
Luego seleccionar el turno:



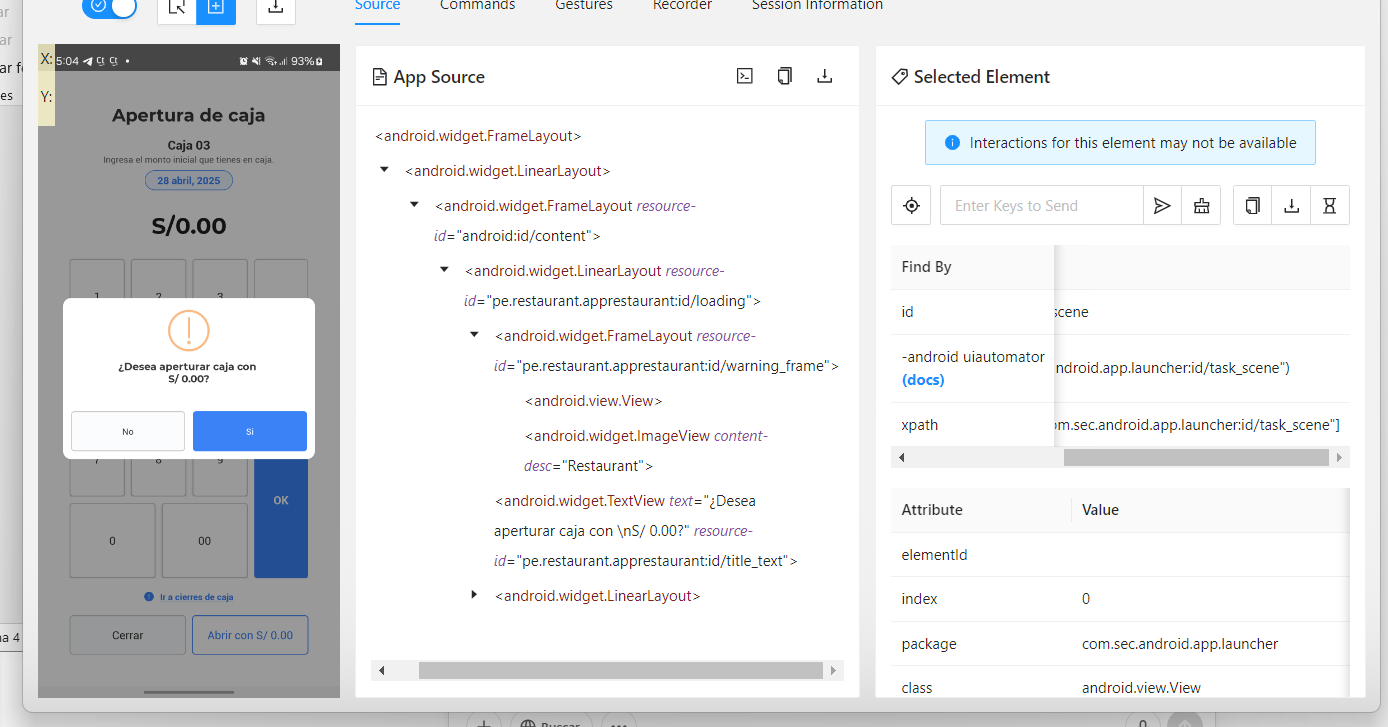
Luego carga el inicio de espera donde se sincronizan los datos de acuerdo a la demo que se han logueado:



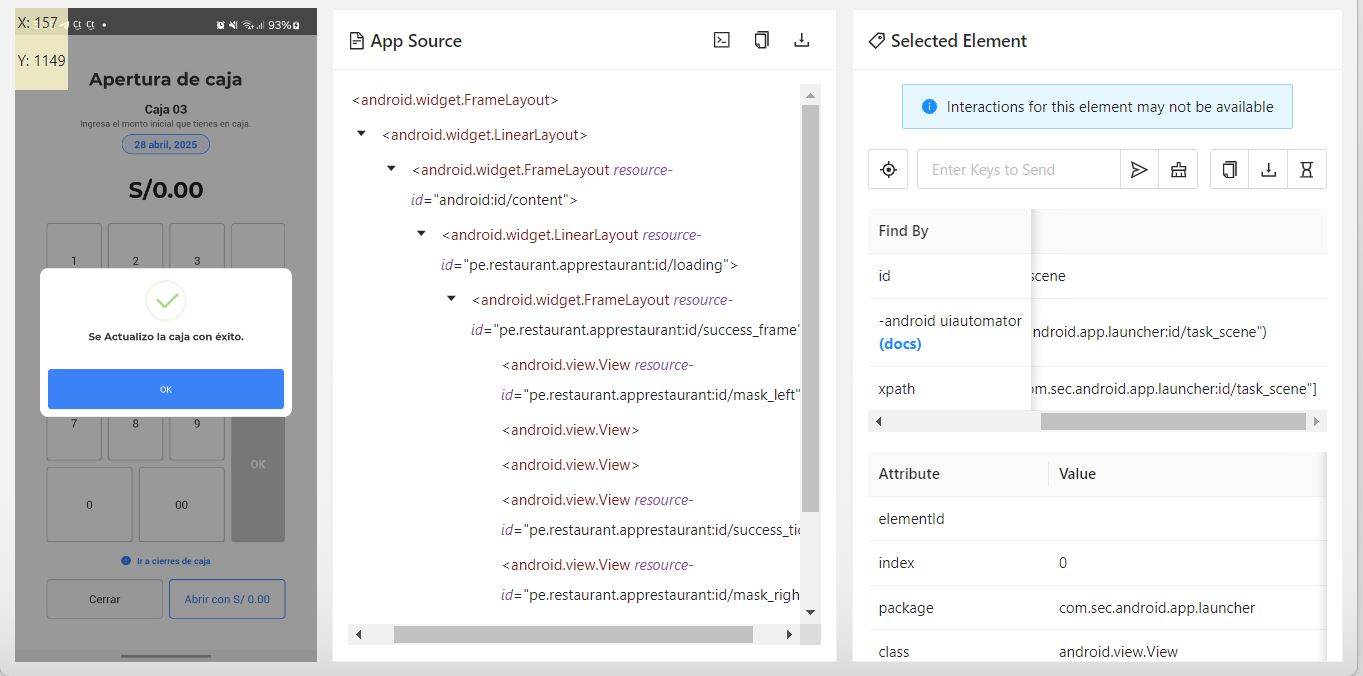
Luego aperturamos caja, ya que la caja debe estar cerrada:



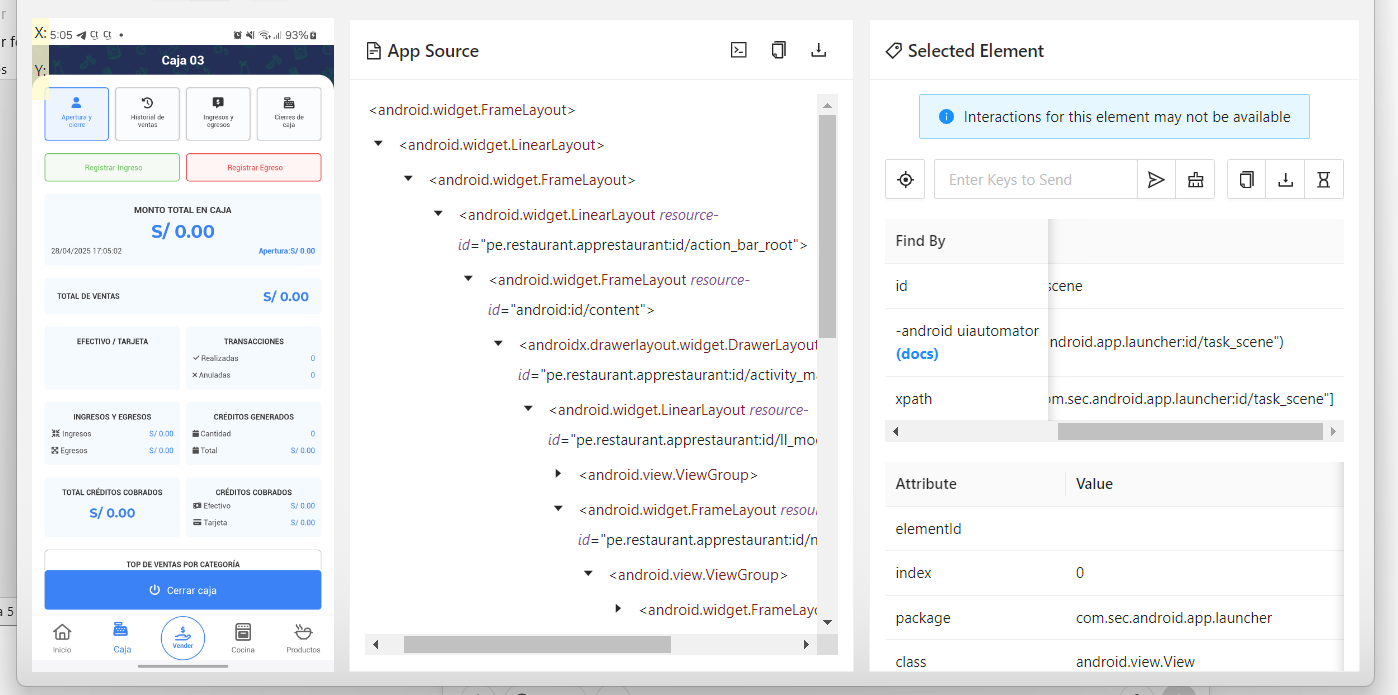
Confirmamos que si quieremos abrirla en cero:



Después confirmamos que si queremos aperurtar caja:

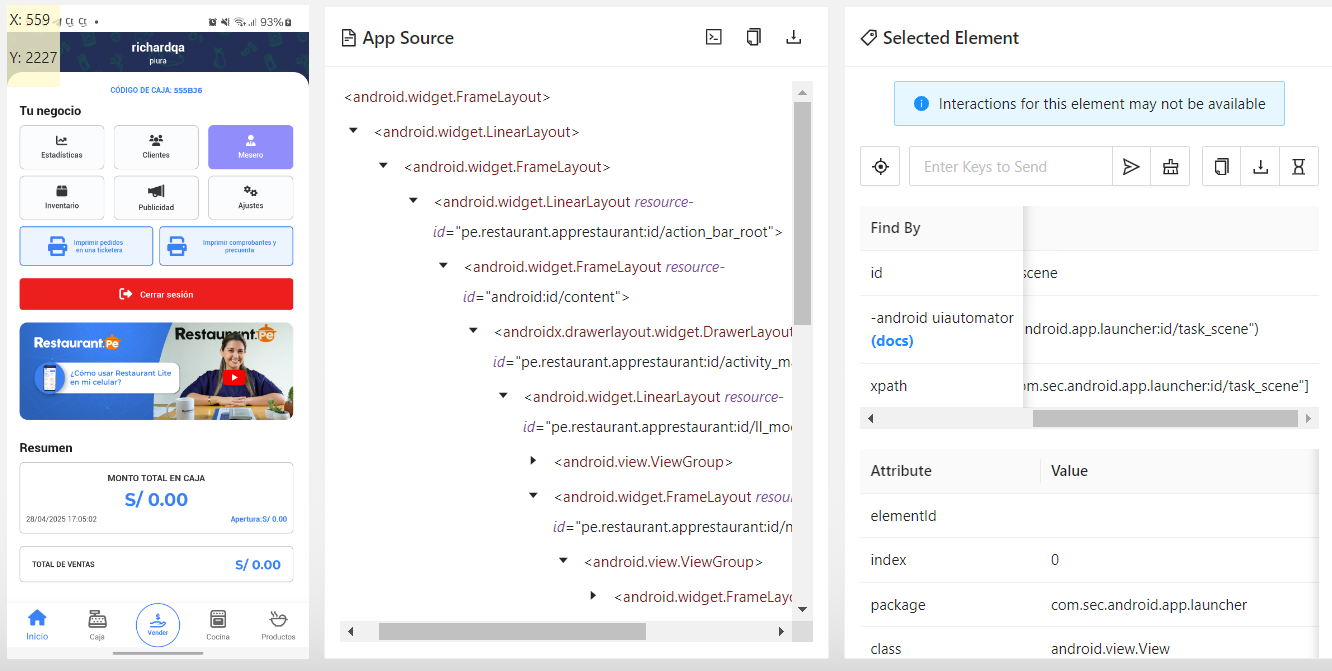


Se apertura el sistema listo para vender:



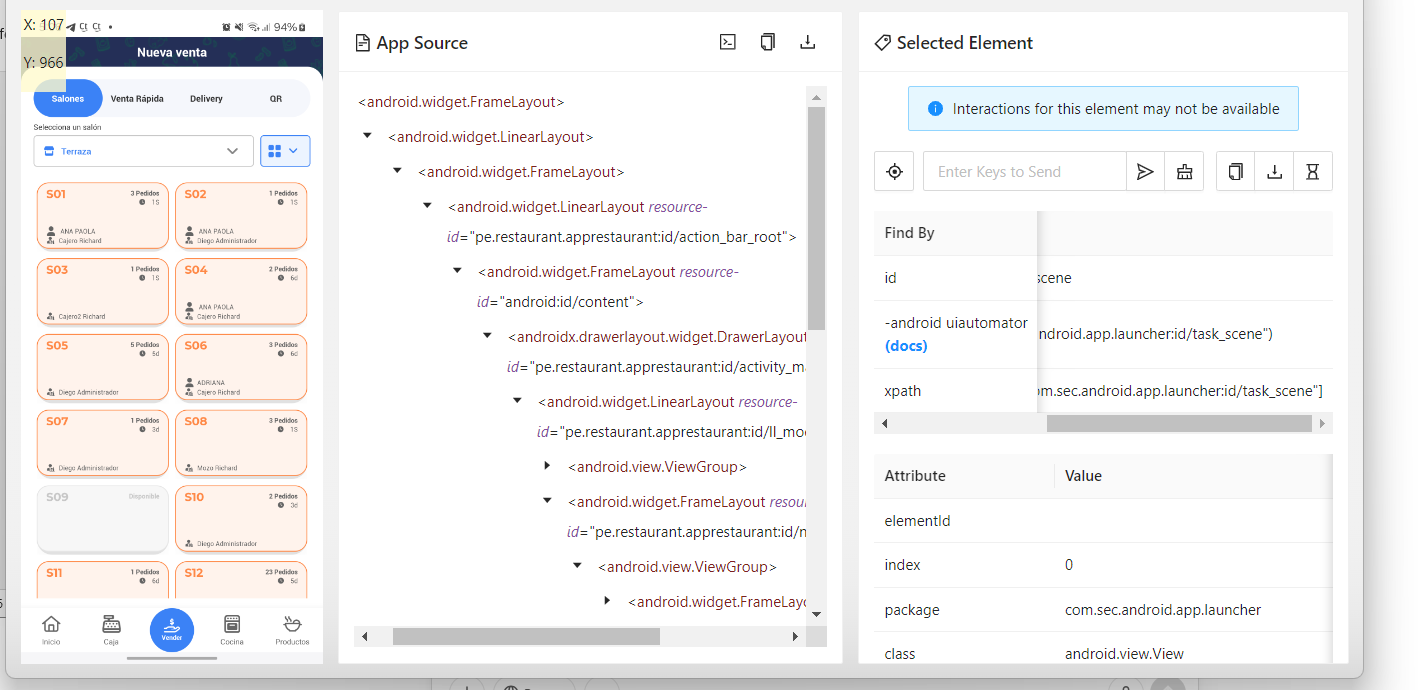
UNA VEZ DENTRO, VAMOS HACER LOS FLUJOS DE VENTAS:

HACEMOSE CLIC EN “VENDER”

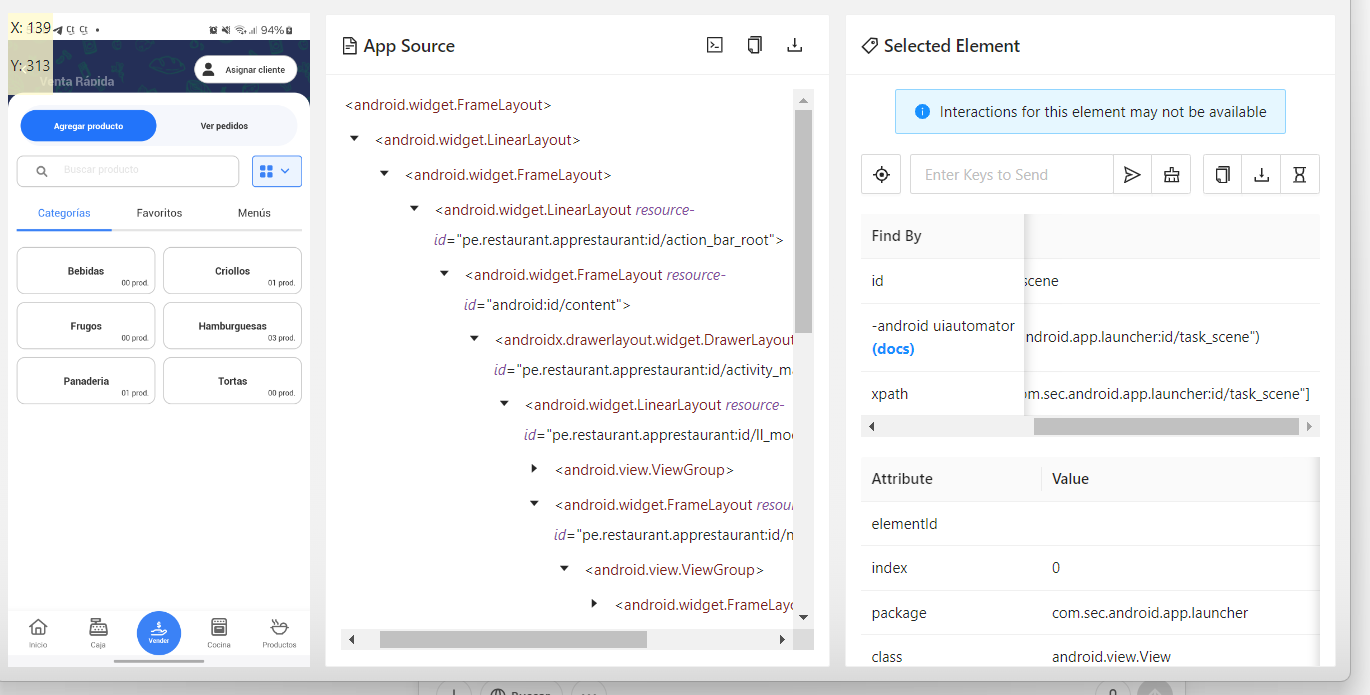


SE MMUESTRA POR DEFECTO LA SELECCIÓN DE SALONES PARA EL FLUJO VENTA POR SALÓN, PERO VAMOS HA REALIZAR VENTA RAPIDA:

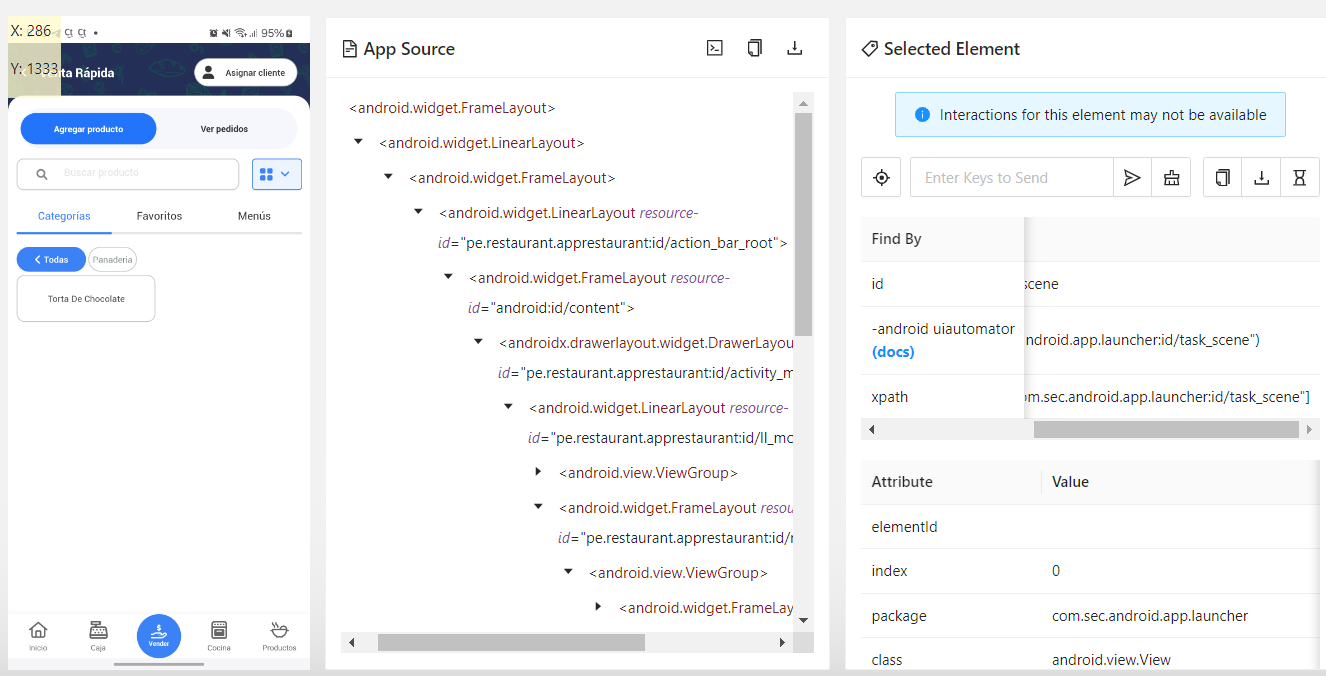
**HACEMOS CLIC EN “VENTA RAPIDA”**



CARGA LA OPCIÓN DE VENTA RAPIDA, AQUÍ VENDEMOS DE MANERA DIRECTA EN CAJA, “CUANDO ALGUEIN VA DE PASADA Y QUIERE ALGO RAPIDO SIN SENTARSE A DEGUSTAR”, TIENE LA BARRA DE BUSQUEDA, LUEGO TENEMOS, CATEGORIA, FAVORITOS Y MENUS, PARA BUSCAR EL PRODUTO QUE SE QUIERE DESPACHAR.



SELECIONMOS UNA CATEGORIA, “PANADERIA”

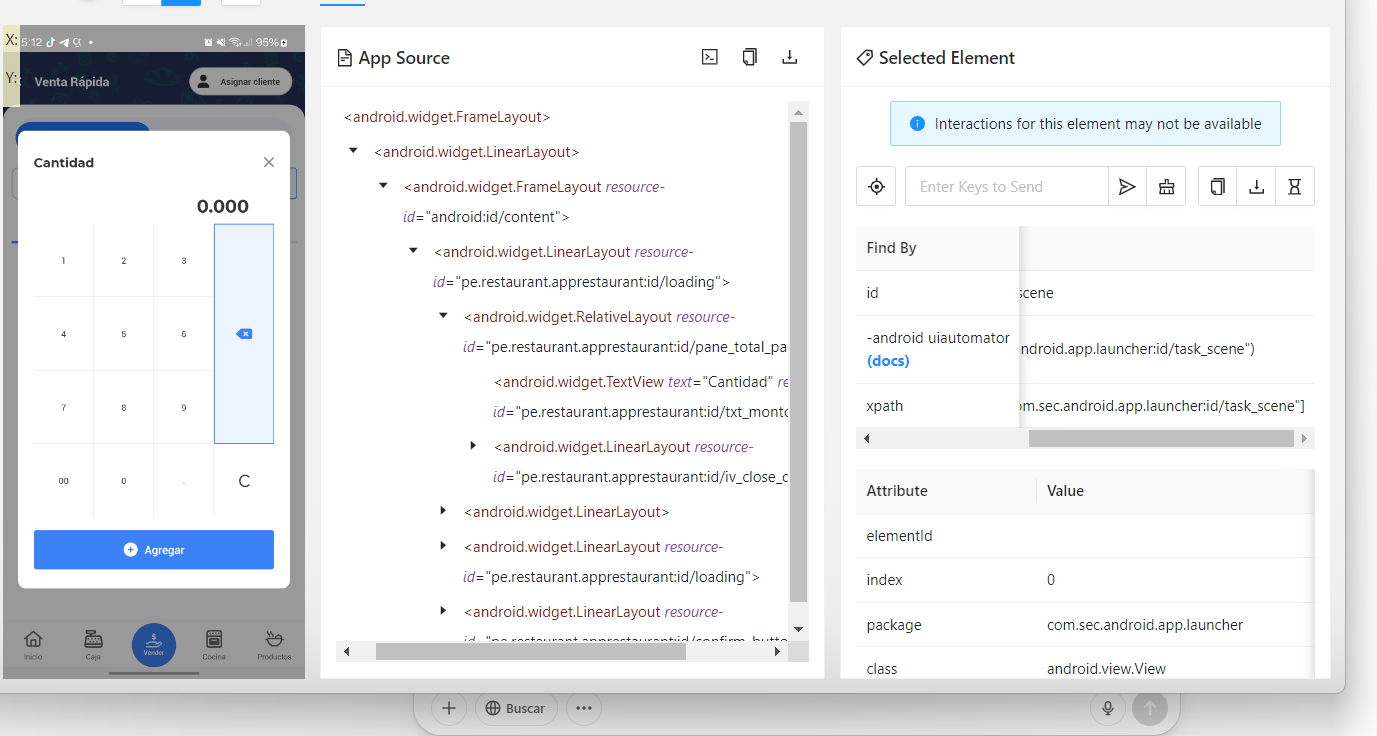


LUEGO HACEMOS CLIC SOBRE LA SUBCATEGORIA TORTA DE CHOCOLATE



LUEGO HACMEOS CLICK SOBRE CUALQUIERA DE LAS PRESENTACIÓNES :

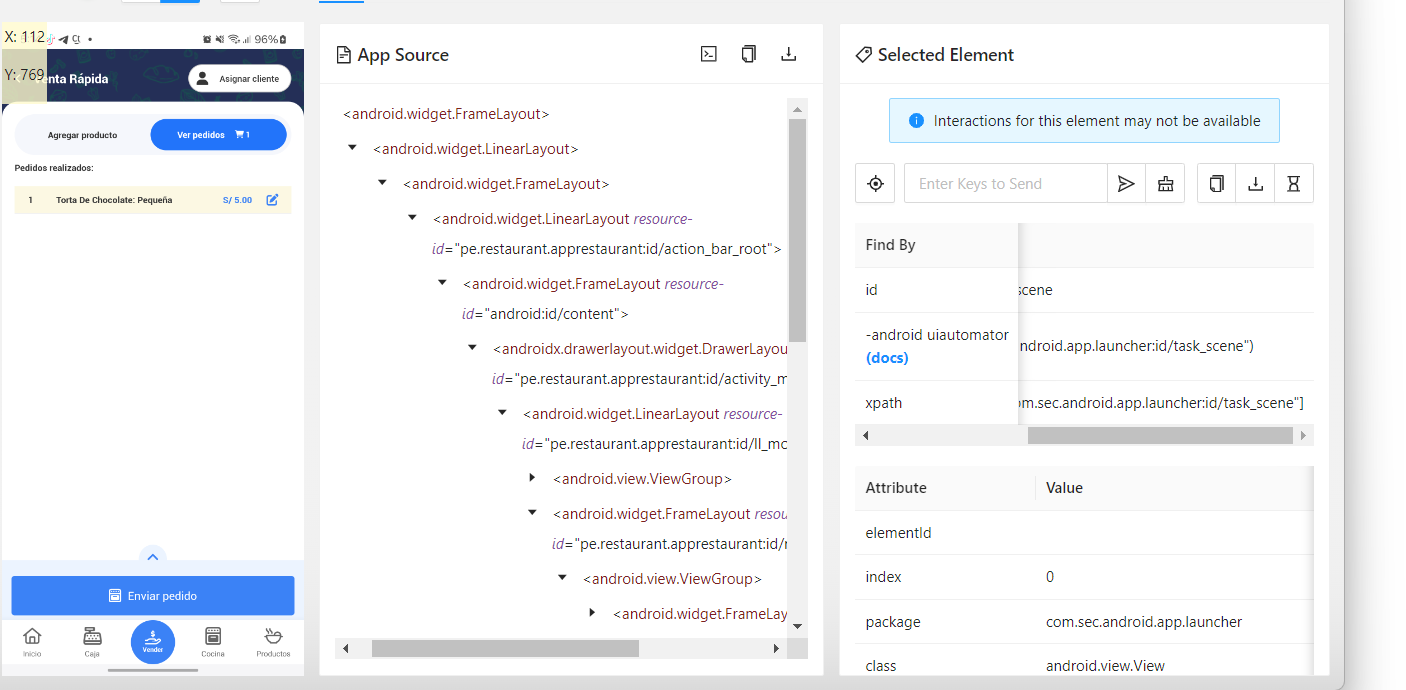
AGREGAMOS LA CANTIDAD QUE VAMOS A LLEVAR Y LE DAMOS EN EL BOTÓN AGREGAR:



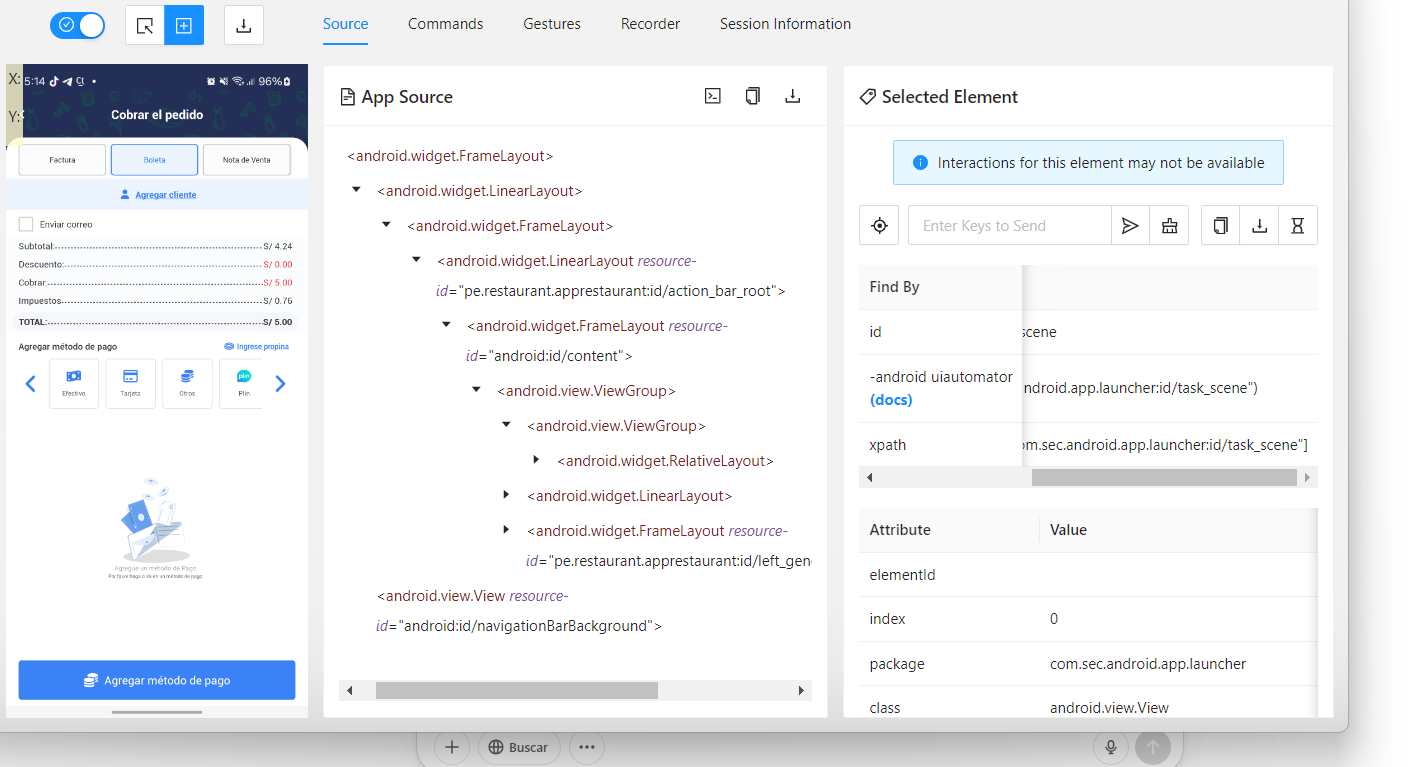
LUEGO SE HACE CLIC EN “VER PEDIDOS”



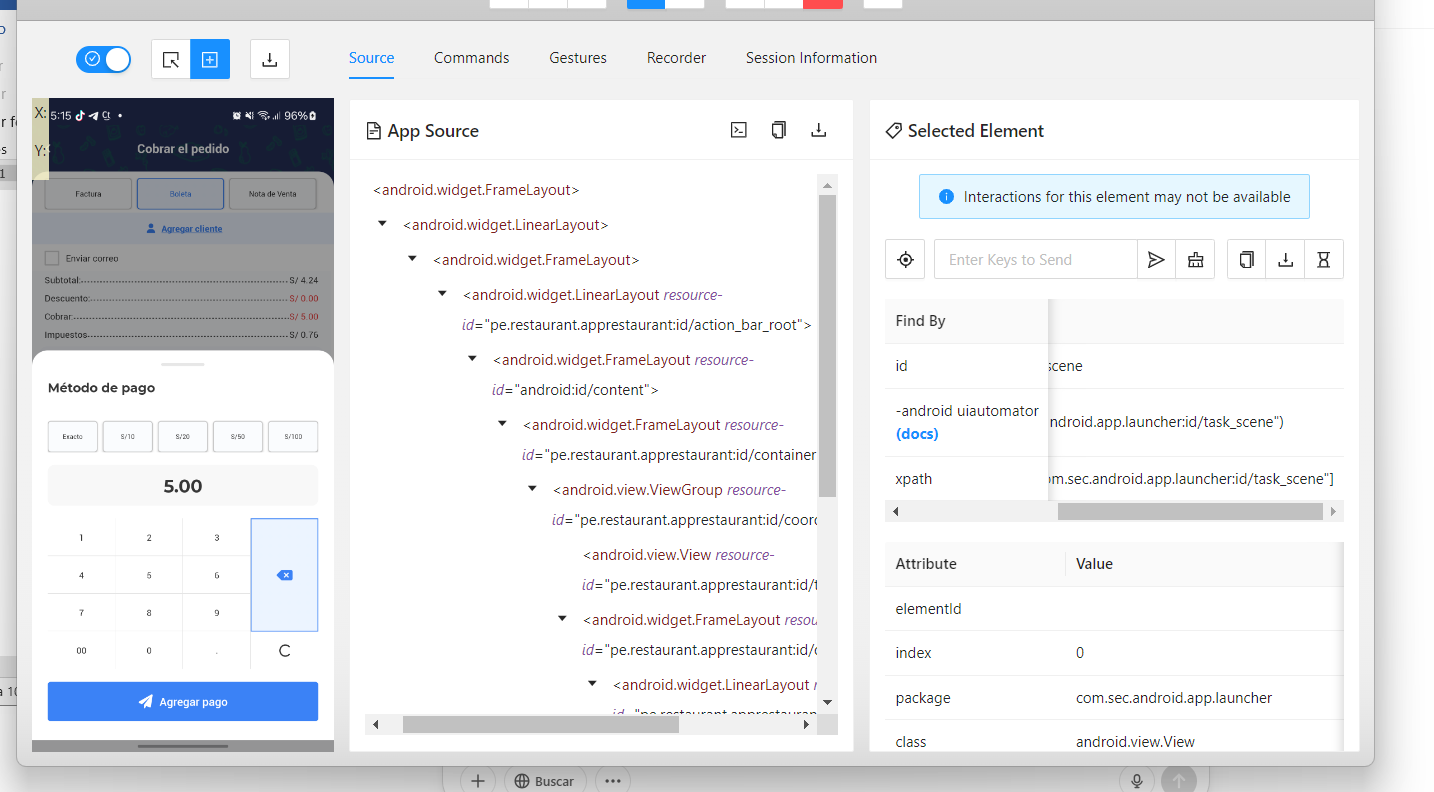
LUEGO HCEMMOS CLIC EN ENVIAR PEDIDO:



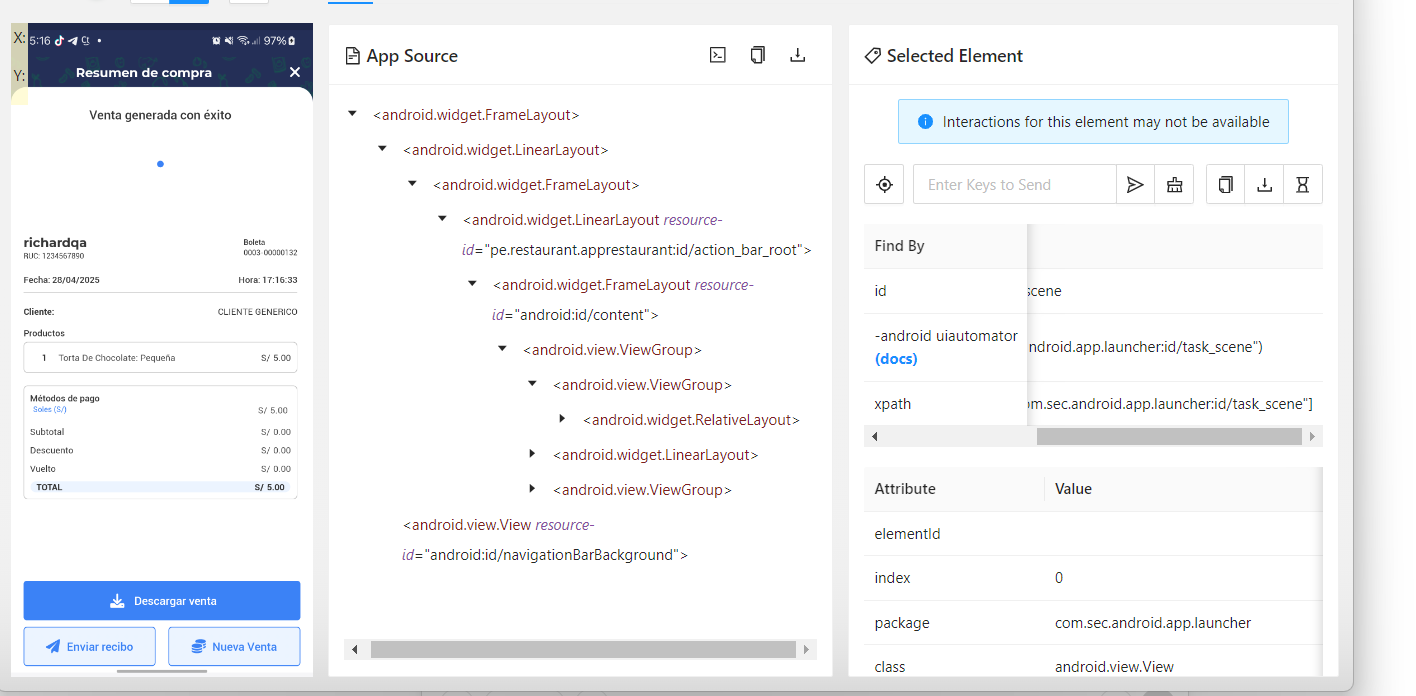
SELECCIONAMOS EL METODO DE PAGO



SELECCINAMOS EFECTIVO, PARA EL EJEMPLO, LUEGO SELECCIOMOS EL MONTO CON EL QUE PAGAN, SI ES EXCATO O LAS OPCIONES QUE SE MUESTRAN, SELECCIONAMOS “EXACTO”:



SE REALIZA AUTOMATICAMENTE LA VENTA:



SI NO ESTAS CONTECTADO A UNA IMPRESORA LUEGO DE UN MOMENTO, EXACTAMENTE MEDIO MINUTO A MÁS , ENVIA UNA NOTIFICACIÓN QUE NO SE IMPRIMIÓ.

LUEGO CERRAMOS LA VENTANA, HACIENDO CLIC EN LA X, O DONDE DICE “NUEVA VENTA”



**ESTRUCTURA DEL PROYECTO EN VS CODE**

**📁 appium\_proyect/**

**│**

**├── 📁 test/ # Scripts de prueba y flujos principales**

**│ ├── appium\_driver.py # Inicializa Appium Driver**

**│ ├── test\_flows.py # Suite de ejecución**

**│ ├── flujo\_venta\_salon.py # Flujo: Venta por salón**

**│ ├── flujo\_cobro\_pedido.py # Flujo: Cobro de pedido**

**│ └── ...**

**│**

**├── 📁 pages/ # Mapeo de pantallas (Page Object Model)**

**│ ├── login\_page.py**

**│ ├── caja\_page.py**

**│ ├── turno\_page.py**

**│ └── ...**

**│**

**├── 📁 clientes/ # Funciones de cliente (registro, edición)**

**│ ├── registro\_cliente\_basico.py**

**│ ├── buscar\_cliente.py**

**│ └── ...**

**│**

**├── 📁 reportes/ # Reportes generados por Allure**

**│ ├── allure-results/**

**│ └── allure-report/**

**│**

**├── 📁 evidencia/ # Capturas de pantalla o logs de errores**

**│ ├── caja\_seleccionada.png**

**│ └── flujo\_caja\_aperturada.png**

**│**

**├── requirements.txt # Lista de dependencias**

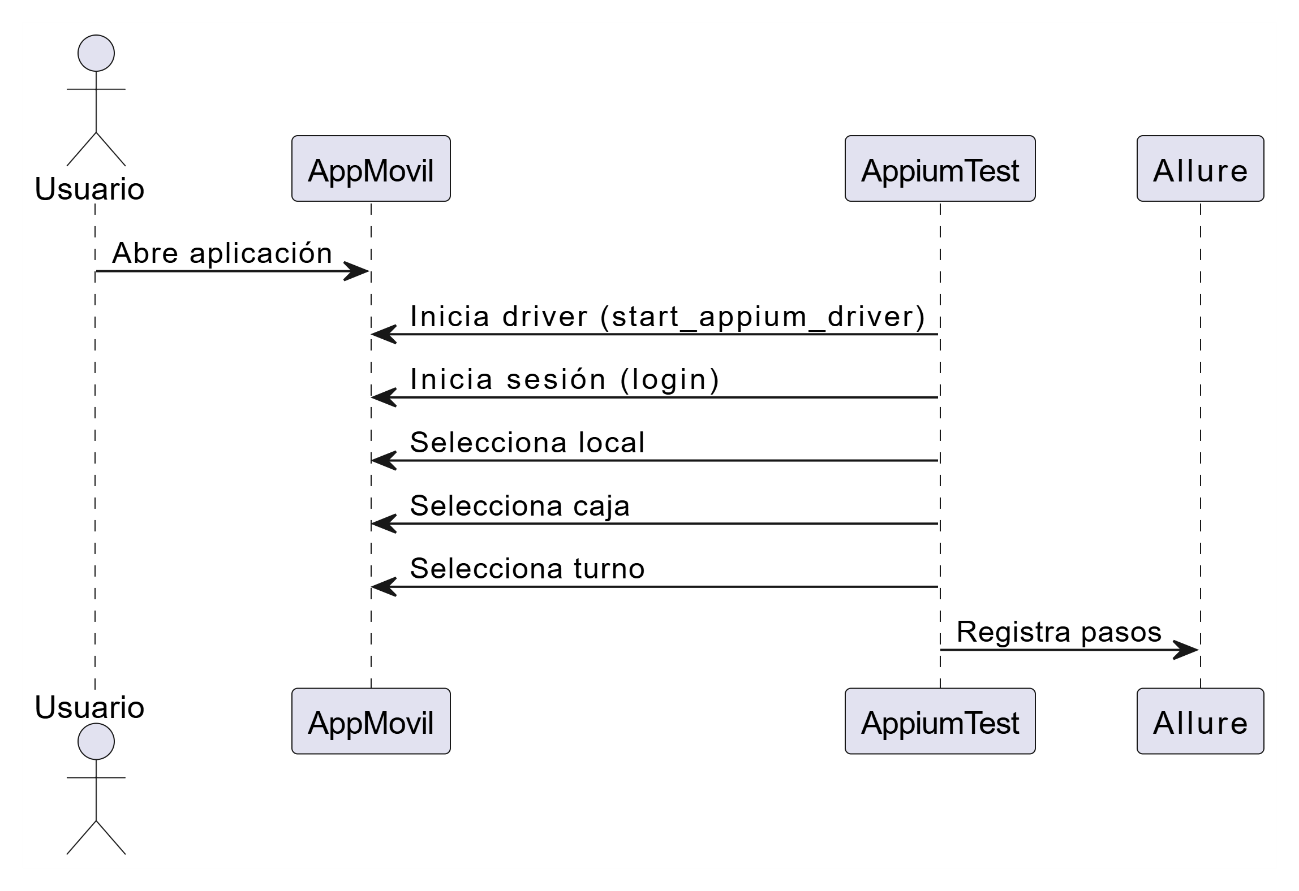
**├── README.md # Descripción del proyecto**

**├── .gitignore # Exclusiones de Git**

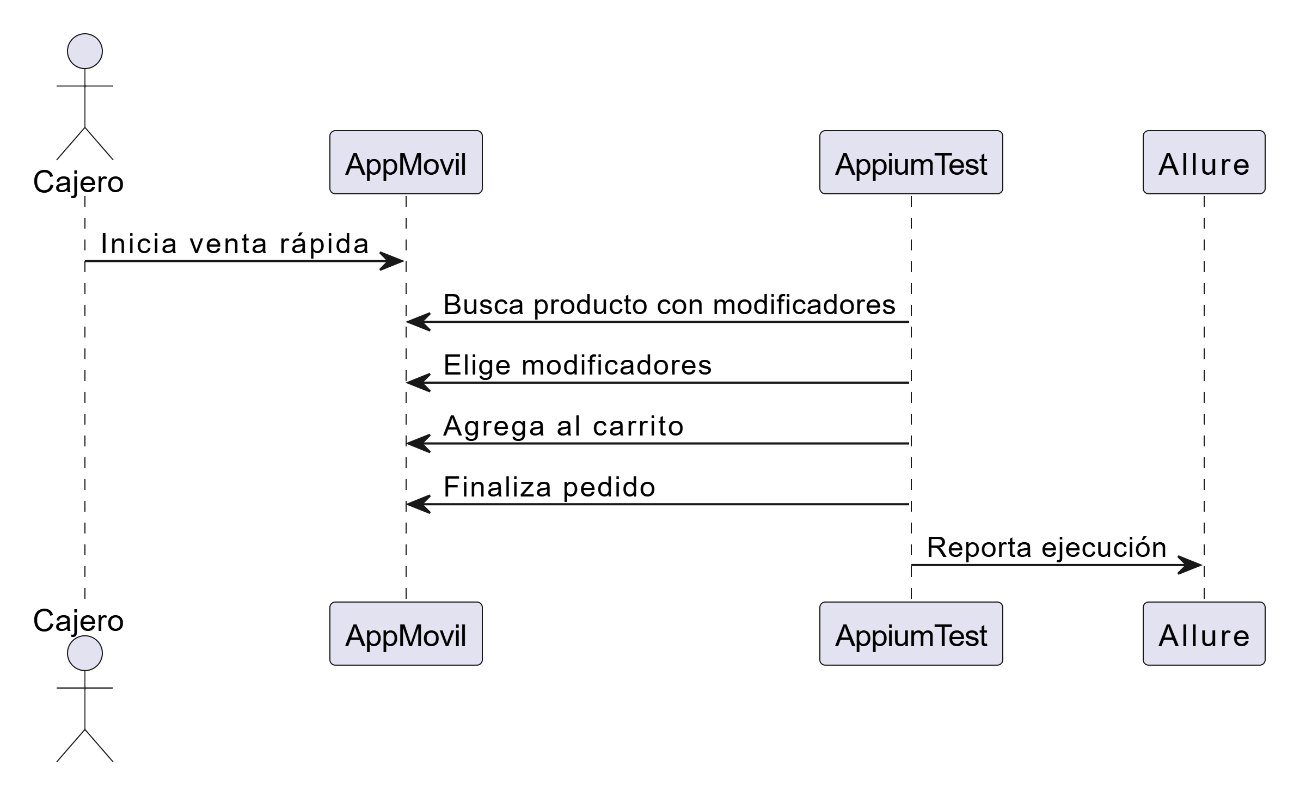
**└── pytest.ini # Configuración de Pytest (si aplica)**

**DIAGRAMAS SE SECUENCIA:**

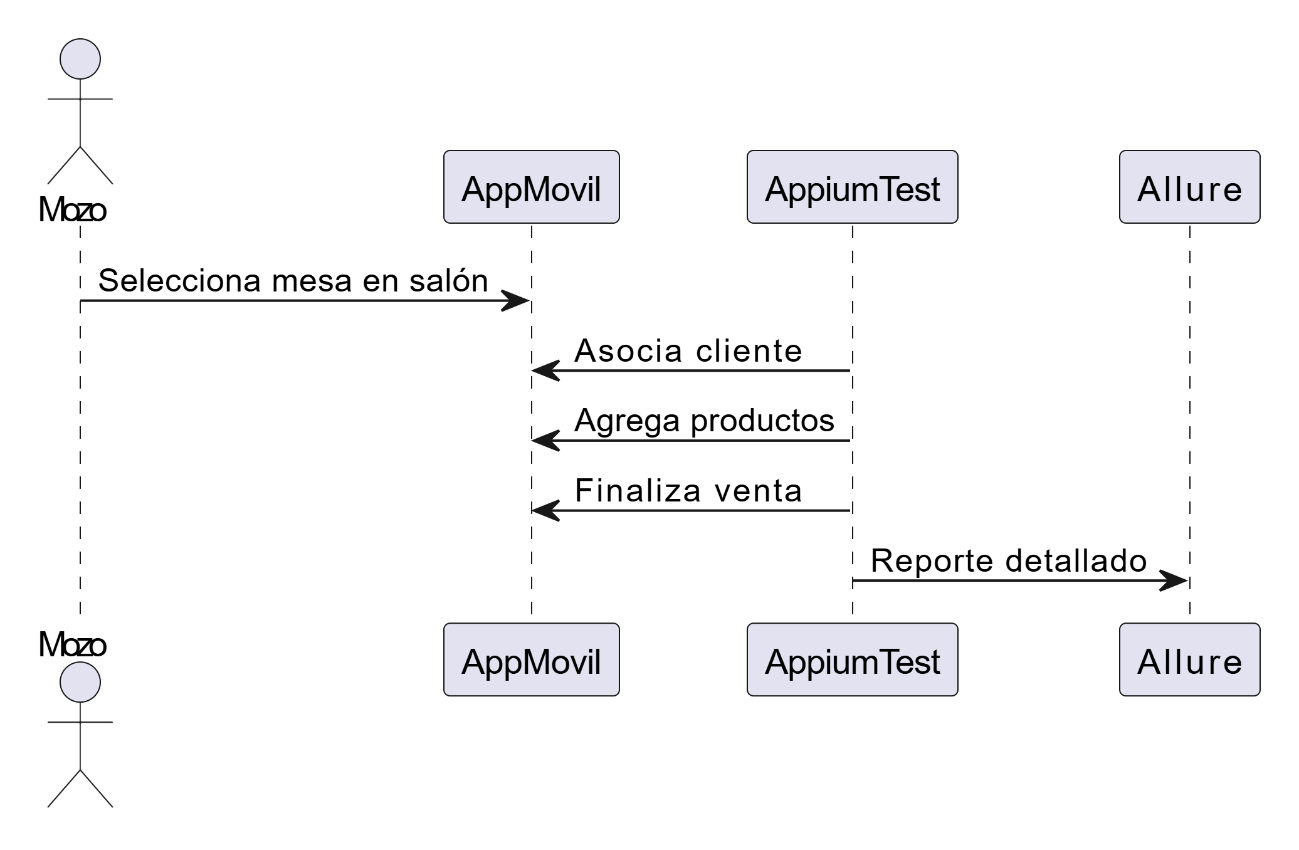
**Login y caja**

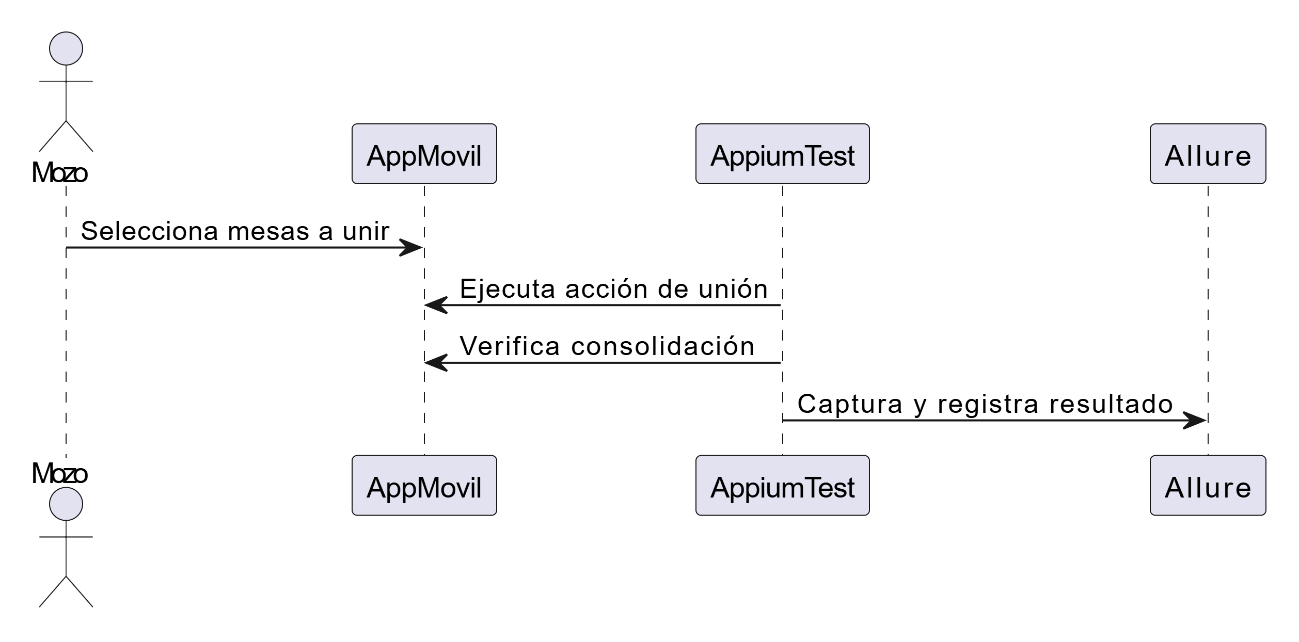
****

**Venta Rápida con Modificadores**

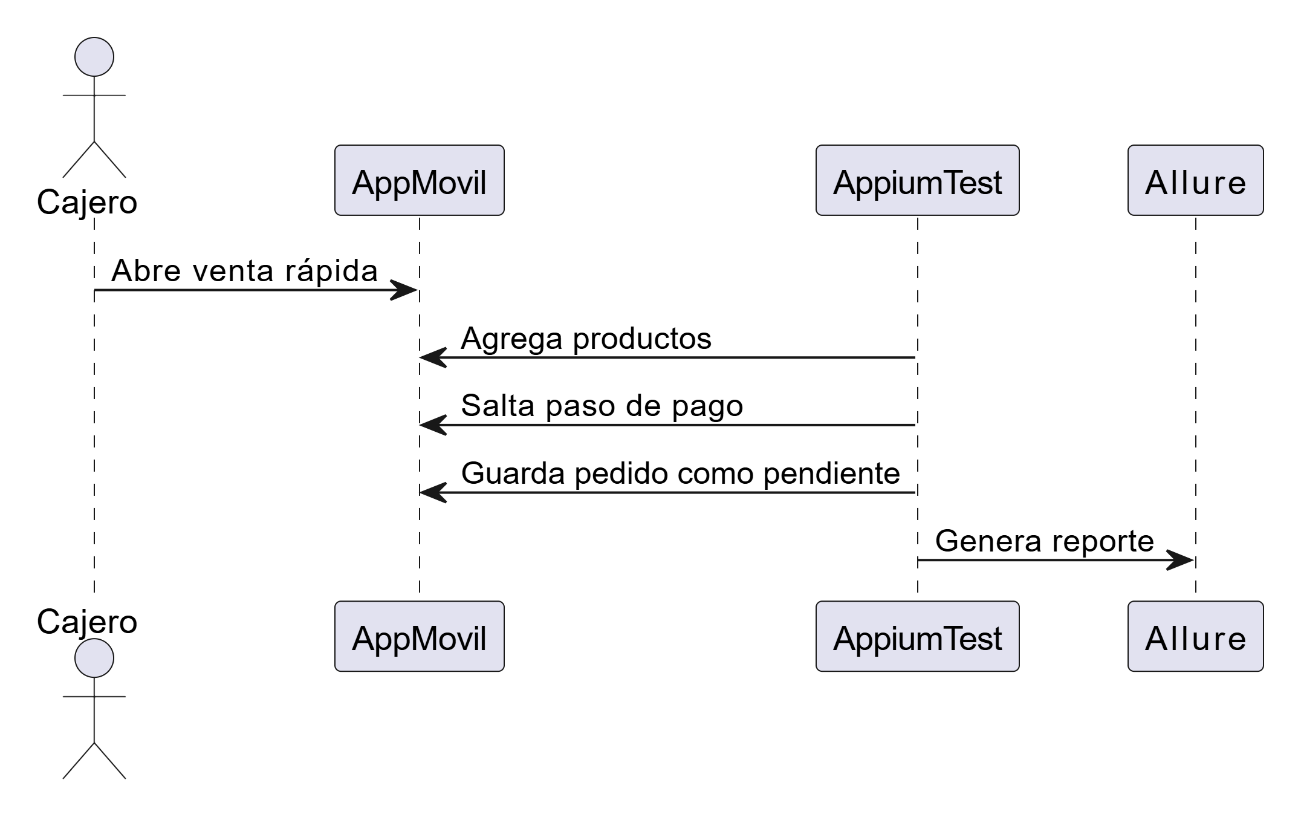
****

**Venta por salón con cliente:**

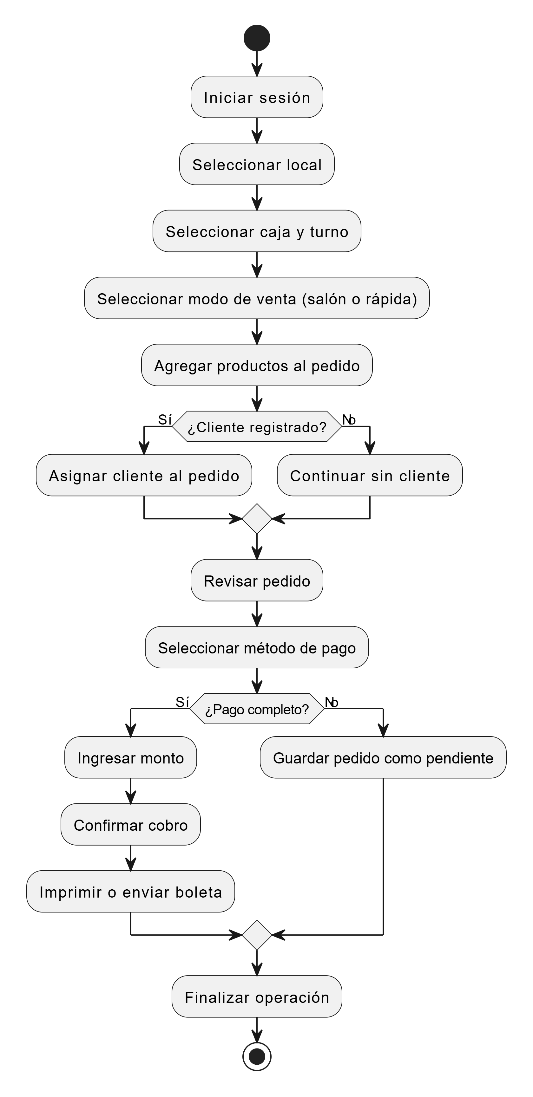
**unión de mesas:**

****

**Venta sin pago:**

****

**DIAGRAMA DE ACTIVIDADES COBRO PEDIDO:**

****

Documentación Técnica de Pruebas Funcionales - App Móvil restaurant.pe

# Resumen del Proyecto

Este proyecto automatiza pruebas funcionales sobre una aplicación móvil del sistema de ventas de restaurant.pe usando Appium Inspector y Client. Permite validar operaciones críticas como login, ventas por salón, ventas rápidas, manejo de turnos y cajas, y generación de reportes en Allure con trazabilidad de cada paso ejecutado.

# Herramientas Utilizadas

- Appium Inspector  
- Appium Client (Python)  
- Pytest / Unittest  
- Allure  
- Emulador Android o dispositivo real

# Casos de Prueba y Flujos

## Inicio y Login de Usuario

- Código:  
start\_appium\_driver() + login()

- Argumento:  
Inicia Appium, conecta el dispositivo, autentica al usuario.

## Selección del Local

- Código:  
seleccionar\_local(driver)

- Argumento:  
El usuario elige el local de atención donde operará.

## Apertura de Caja

- Código:  
seleccionar\_caja(driver)

- Argumento:  
Simula el inicio de caja antes de registrar ventas.

## Selección del Turno

- Código:  
seleccionar\_turno(driver)

- Argumento:  
El cajero selecciona su turno operativo.

## Venta rápida con modificadores

- Código:  
flujo\_venta\_rapida\_con\_modificadores(driver)

- Argumento:  
Agrega productos con personalización (ej. sin cebolla).

## Venta rápida con cantidades

- Código:  
flujo\_venta\_rapida\_con\_cantidades(driver)

- Argumento:  
Permite seleccionar múltiples unidades de un producto.

## Venta desde favoritos

- Código:  
flujo\_venta\_rapida\_desde\_favoritos(driver)

- Argumento:  
Agrega productos frecuentes desde favoritos.

## Venta por salón con cliente

- Código:  
venta\_por\_salon\_con\_cliente(driver)

- Argumento:  
Simula venta en mesa física asignando cliente.

## Unión de mesas

- Código:  
venta\_por\_salon\_union\_mesas(driver)

- Argumento:  
Permite consolidar pedidos de dos mesas.

## Mover pedido entre mesas

- Código:  
venta\_por\_salon\_mover\_pedidos(driver)

- Argumento:  
Transfiere un pedido de una mesa a otra.

## Venta rápida sin pagar

- Código:  
flujo\_venta\_rapida\_sin\_pagar(driver)

- Argumento:  
Guarda pedidos pendientes sin pago inmediato.

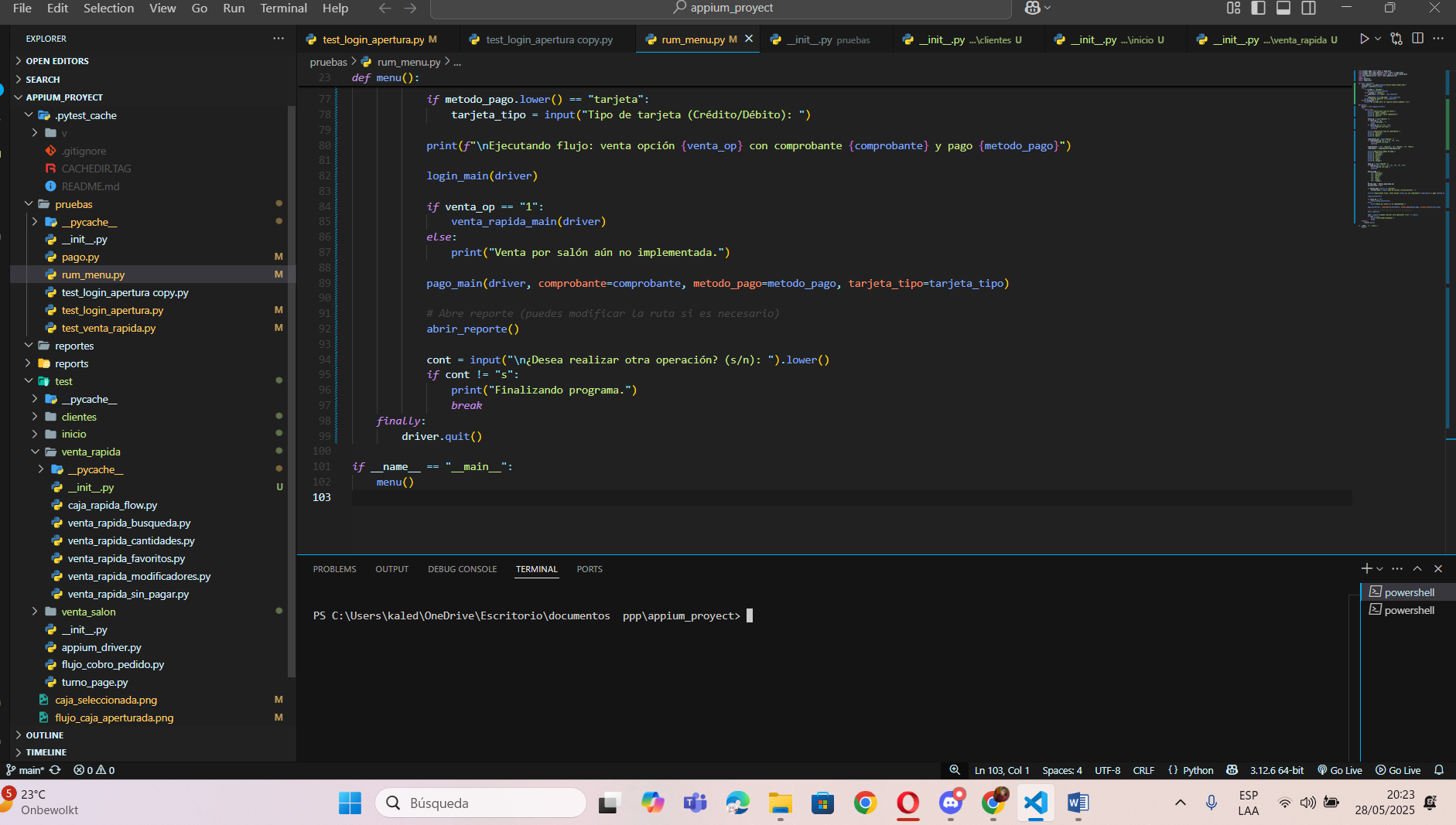
## Búsqueda de producto

- Código:  
venta\_rapida\_busqueda(driver)

- Argumento:  
Busca y selecciona productos por nombre.

# Evidencias Visuales

Evidencia: Caja seleccionada.png



*from* pruebas.pago *import* main *as* pago\_main

*from* pruebas.test\_login\_apertura *import* main *as* login\_main

*from* pruebas.test\_venta\_rapida *import* main *as* venta\_rapida\_main

*from* test.appium\_driver *import* start\_appium\_driver

*import* os

*import* platform

*import* subprocess

*def* abrir\_reporte():

    ruta\_reporte = os.path.abspath("allure-report/index.html")

    sistema = platform.system()

*try*:

*if* sistema == "Windows":

            os.startfile(ruta\_reporte)

*elif* sistema == "Darwin":

            subprocess.run(["open", ruta\_reporte])

*else*:

            subprocess.run(["xdg-open", ruta\_reporte])

        print(*f*"Reporte abierto: {ruta\_reporte}")

*except* Exception *as* e:

        print(*f*"No se pudo abrir el reporte automáticamente: {e}")

*def* menu():

    driver = start\_appium\_driver()

*try*:

*while* True:

            print("\nSeleccione tipo de venta:")

            print("1. Venta rápida")

            print("2. Venta por salón (pendiente)")

            print("3. Salir")

            venta\_op = input("Opción: ")

*if* venta\_op == "3":

                print("Saliendo...")

*break*

*if* venta\_op not in ["1", "2"]:

                print("Opción inválida.")

*continue*

            print("\nSeleccione tipo de comprobante:")

            print("1. Factura")

            print("2. Boleta")

            print("3. Nota")

            comprobante\_op = input("Opción: ")

*if* comprobante\_op not in ["1", "2", "3"]:

                print("Opción inválida.")

*continue*

            comprobantes = {"1": "Factura", "2": "Boleta", "3": "Nota"}

            comprobante = comprobantes[comprobante\_op]

            print("\nSeleccione medio de pago:")

            print("1. Efectivo")

            print("2. Tarjeta")

            print("3. Otros")

            print("4. Plin")

            print("5. Yape")

            print("6. Izipay")

            pago\_op = input("Opción: ")

*if* pago\_op not in ["1", "2", "3", "4", "5", "6"]:

                print("Opción inválida.")

*continue*

            medios\_pago = {

                "1": "Efectivo",

                "2": "Tarjeta",

                "3": "Otros",

                "4": "Plin",

                "5": "Yape",

                "6": "Izipay",

            }

            metodo\_pago = medios\_pago[pago\_op]

            tarjeta\_tipo = None

*if* metodo\_pago.lower() == "tarjeta":

                tarjeta\_tipo = input("Tipo de tarjeta (Crédito/Débito): ")

            print(*f*"\nEjecutando flujo: venta opción {venta\_op} con comprobante {comprobante} y pago {metodo\_pago}")

            login\_main(driver)

*if* venta\_op == "1":

                venta\_rapida\_main(driver)

*else*:

                print("Venta por salón aún no implementada.")

            pago\_main(driver, comprobante=comprobante, metodo\_pago=metodo\_pago, tarjeta\_tipo=tarjeta\_tipo)

*# Abre reporte (puedes modificar la ruta si es necesario)*

            abrir\_reporte()

            cont = input("\n¿Desea realizar otra operación? (s/n): ").lower()

*if* cont != "s":

                print("Finalizando programa.")

*break*

*finally*:

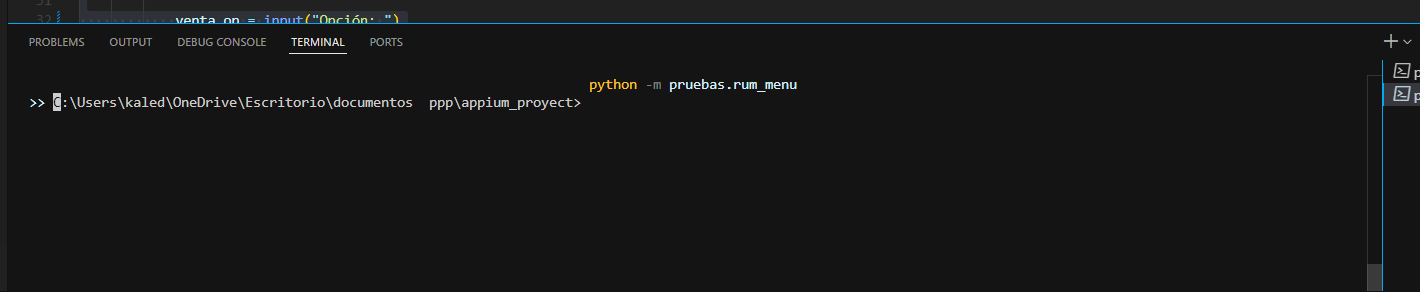
        driver.quit()

*if* \_\_name\_\_ == "\_\_main\_\_":

    menu()

REPORTES:

EJECUTAMOS:



REALIZA LA EJECUCIÓN EN EL MOVIL Y AL TERMINAR ENVIA UN REPORTE EN ALLURE:

