Appium Grid con **Selenium Grid 4**

“Este es un ejemplo de ejecución para Android y iOS ejecutando un comportamiento en la aplicación ‘YouTube’ sobre el ejemplo/base publicado en el Proyecto TESTINCONTINUOU de GitLab Umane”

1. Herramientas como pre-requisitos
   - Tener instalado java 11 y configurada la variable de ambiente JAVA_HOME
   - Tener instalado maven y configurada la variable de ambiente M2_HOME
   - Tener instalado el SDK Android y configurada la variable de ambiente ANDROID_HOME
   - Tener instalada la útlima versión (1.22.3) de appium por comandos
   - (Solo iOS) Tener instalado Xcode

2. Instalar la librería ***mobile-automation-0.0.7-SNAPSHOT***
   - Clonar librería de la rama develop
     $ git clone -b develop https://umane.everis.com/git/TESTCONTINUOEVERIS/everis-lib-mobile-appium-java.git
   - Instalar librería en .m2
     $ cd <repositorio-clonado>
     $ mvn clean install

3. Clonar proyecto ejemplo
   - Clonar repositorio de la rama master
     $ git clone https://umane.everis.com/git/TESTCONTINUOEVERIS/everis-app-mobile-appium-java.git
   - Validar que en el archivo pom.xml se esté apuntando hacia la última versión del artefacto ***mobile-automation-0.0.7-SNAPSHOT***

```
...
<dependencies>
    <dependency>
        <groupId>com.everis.continuous-testing</groupId>
        <artifactId>mobile-automation</artifactId>
        <version>0.0.7-SNAPSHOT</version>
    </dependency>
</dependencies>
...
```

4. Descargar el .jar Standalone de Selenium Grid 4 (4.12)
   https://www.selenium.dev/downloads/

5. Crear una archivo con un nombre a elección y con la extension **.toml**,por ejemplo '**miFristConfigConnection.toml'**, y agregaro el siguiente contenido pintado en amarillo:
   - Cambiar el valor de la llave 'browserName' y 'appium:platformVersion'

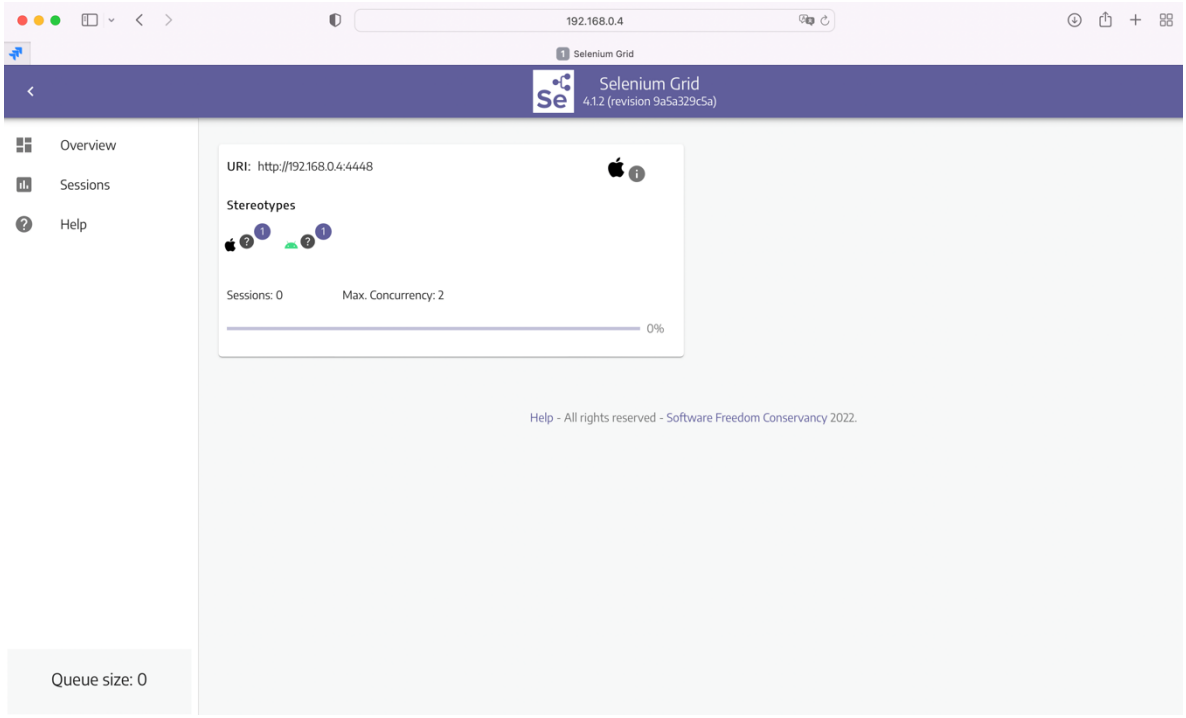
```
[server]
port = 4444

[node]
detect-drivers = false

[relay]
url = "http://localhost:4723/wd/hub"
status-endpoint = "/status"
configs = [
  "1", '{"browserName": "NO_BROWSER", "platformName": "android"}',
  "1", '{"browserName": "NO_BROWSER", "platformName": "iOS"}'
]
```

6. Crear una carpeta con un nombre a elección, por ejemplo '**seleniumGrid**' y pegar el .jar standalone descargado en el punto (4) y el archivo creado en el punto (5).

7. En un terminal/cmd cualquiera ejecutar el comando para crear una instancia del servidor de appium.
   $ appium

8. A nivel de terminal/cmd ir hacia la carpeta creada en el punto (6) y ejecutar el comando para crear la sesión del servidor standalone.
   $ cd /path/directory/**seleniumGrid**
   $ java -jar **selenium-server-4.1.2.jar** standalone --config **config.toml**

   - **Nos aparecera en el mismo terminal el ip del servidor del grid que se acaba de crear, por lo general configurado por el puerto especificado en el archivo.toml en la sección [server] y este tomara la sesión del hub por defecto de appium esfecificado en el archivo.toml en la sección [relay].**
   - **Por ejemplo http://192.168.0.4:4444/**
   - **Copiamos esa ruta y la pegamos en el browser de preferencia, nos aparecerá el siguiente dashboard un nodo instanciado con dos configuraciones en particular: android y ios.**

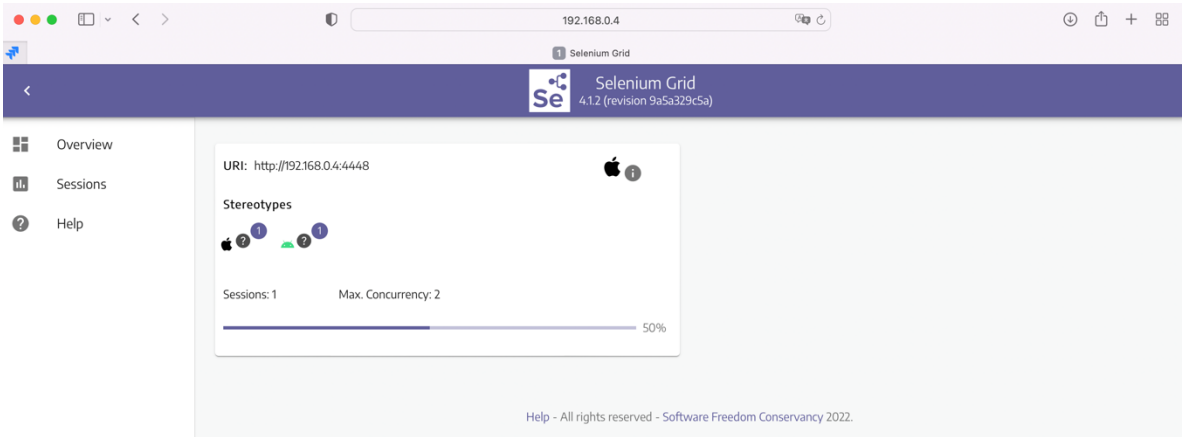


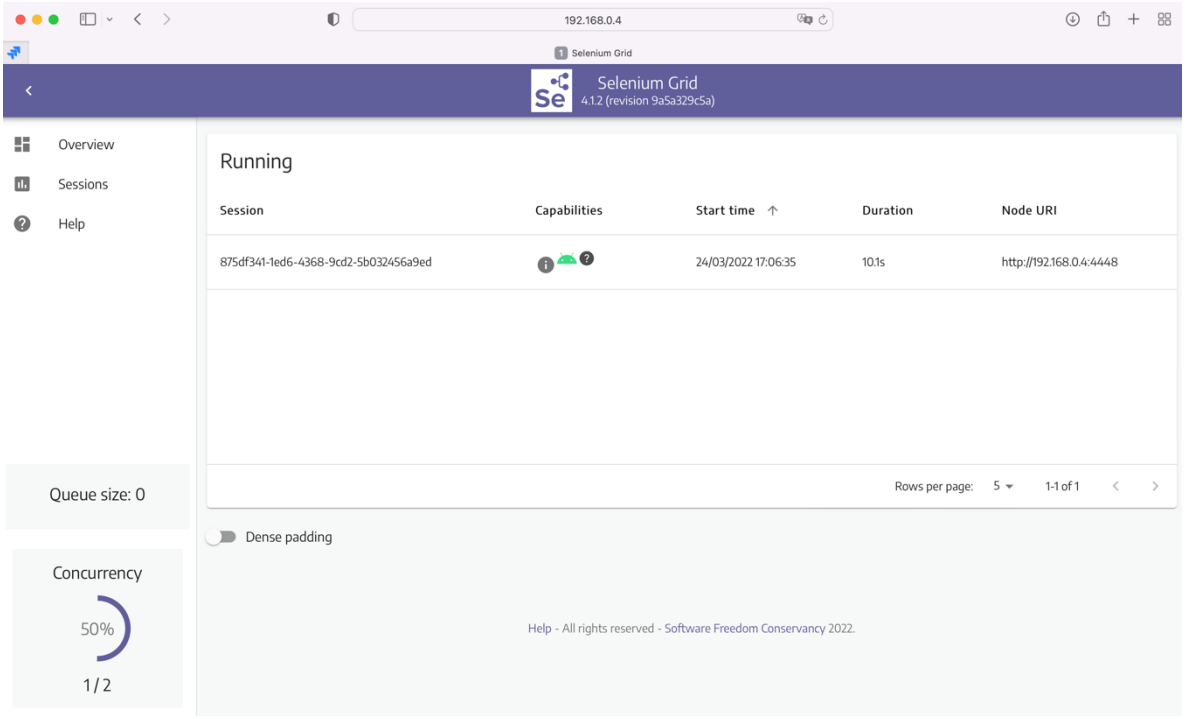
9. Configurar el proyecto ejemplo para realizar la ejecución apuntando a este nuevo "appium-grid"
    - Abrir el archivo de configuración application.yml en el proyecto **mobile-automation-example** y agregar la propiedad 'mobiledriver.appiumGrid' con el valor 'on' o 'true'

      ...
      ```
      mobiledriver:
        implicitWaitOnSeconds: 30
        appiumGrid: on
      ```
      ...
    - Agregar las siguientes propiedades de capabilities en el arhivo **application.yml**: Cambiar los valores por

      ...
      ```
      capabilities:
        appiumHub: http://<ip_grid>:<puerto_grid>
        automationName: UiAutomator2
        platformName: Android
        platformVersion: <version_del_sistemaOperativo>
        deviceName: Mi A3
        app:
        udid: <udid_del_dispositivo>
        android:
          appActivity:
      com.google.android.apps.youtube.app.WatchWhileActivity
          appPackage: com.google.android.youtube
      ```
      ...

10. Ejecutar desde el Junit. Veremos como en el Grid de Selenium se crea un nueva sesión y la sección de sessiones veremos la sesión creada.





11. Configurar ejecucion en paralelo de dispositivos (Android y iOS)
    - Crear un nuevo archivo de configuración de perfiles con los siguientes nombres
      -application-android.yml
      -application-ios.yml

    - Agregar las capabilities necesarias para cada sistema operativo

      Ios:

```
capabilities:
  appiumHub: http://192.168.0.4:4443
  #appiumHub: http://127.0.0.1:4723/wd/hub
  platformName: iOS
  #  platformVersion: 15.4
  config: '{"automationName":"XCUITest",
            "app":"",
            "appium:udid":"00008101-000E69583E44001E",
            "deviceName":"iPhone de Percy",
            "bundleId":"com.google.ios.youtube"}'
```

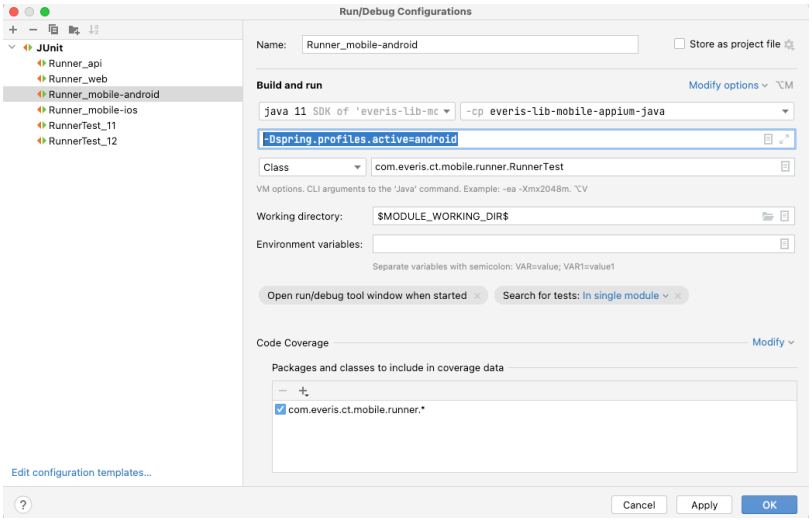
      Android:

```
capabilities:
  appiumHub: http://192.168.0.4:4443
  platformName: android
  #appiumHub: http://127.0.0.1:4723/wd/hub
  config: '{"automationName":"UiAutomator2",
            "app":"",
            "appium:udid":"79eff58d",
            "appPackage":"com.google.android.youtube",

"appActivity":"com.google.android.apps.youtube.app.WatchWhile
Activity"}'
```

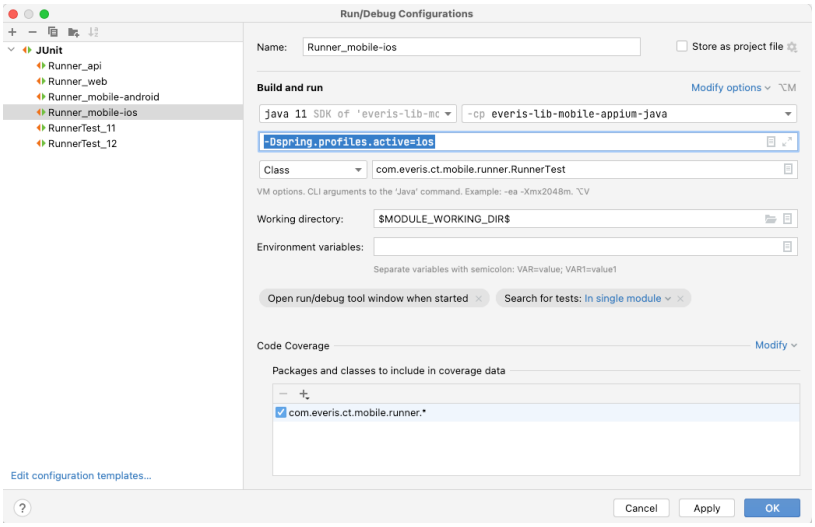
12. Ejecución en paralelo por Junit o commandos.

- Por JUNIT: Crear una ejecución para cada sistema operativo agregandole la propiedad

RunnerTest_android
-Dspring.profiles.active=android



RunnerTest_ios
-Dspring.profiles.active=ios

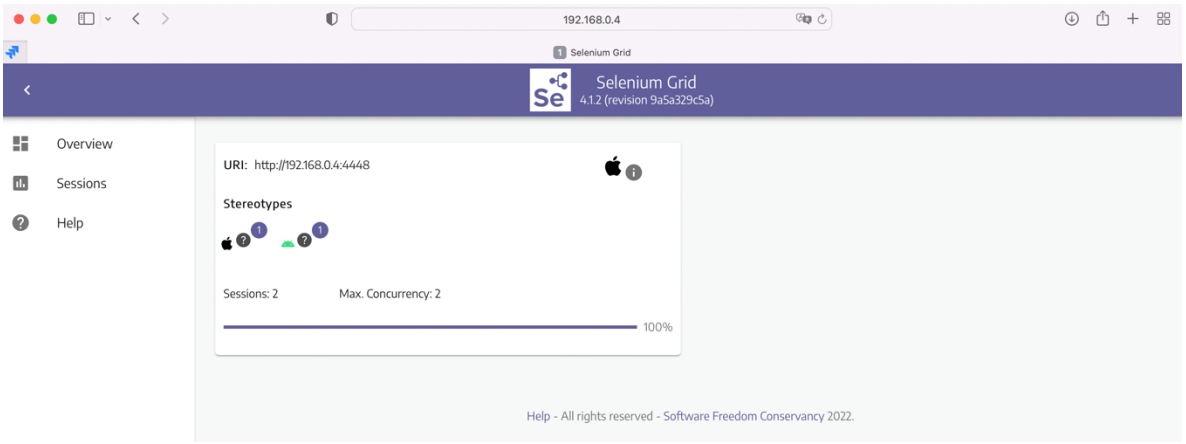


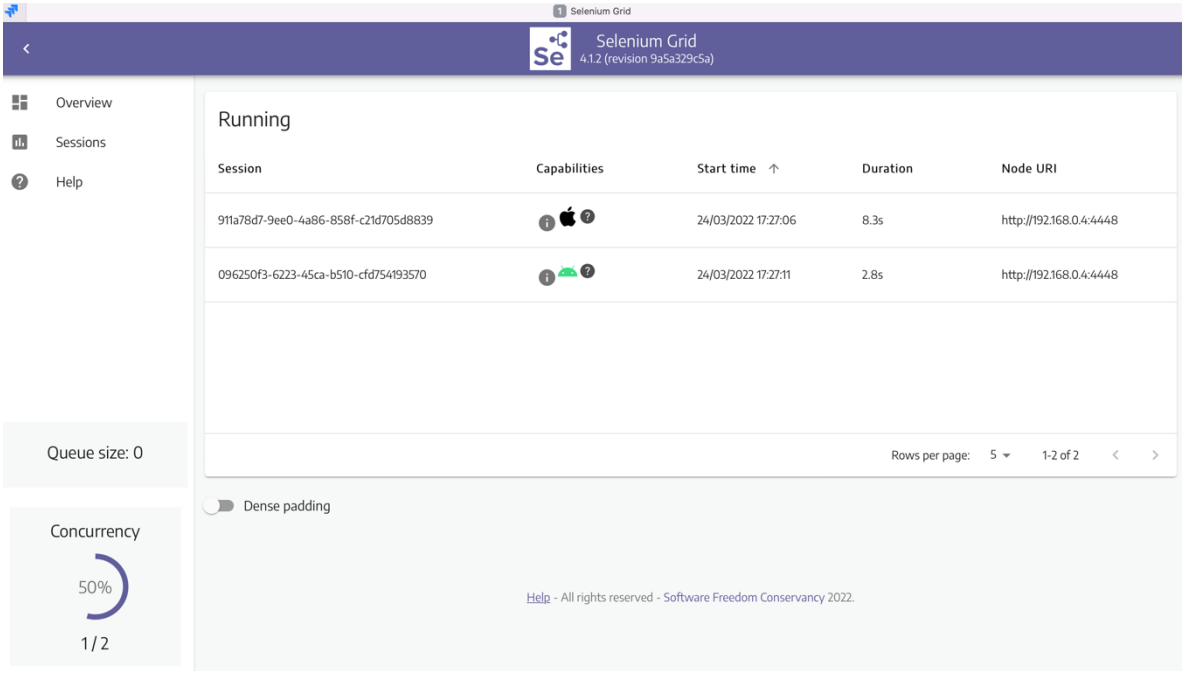
- Por Comandos: Abrir dos terminales independientes y ejecutar en cada uno los siguientes comandos

    - Terminal 1 (Android)
      $ mvn verify **-Dspring.profiles.active=android**

    - Terminal 2 (iOS)
      $ mvn verify **-Dspring.profiles.active=ios**

13. Resultado en DashBoard Selenium Grid





Versión

Responsable	Fecha Modificación	Modificación	Versión
Percy Mendoza	24/03/2022	Primera version	1.0
Percy Mendoza	25/03/2022	Se agregó una nueva capacidad para enviar todo la configuración de capacidades deseadas por json/map	2.0