Let’s wdi5

Creating a wdi5 test scenario in SAP BAS

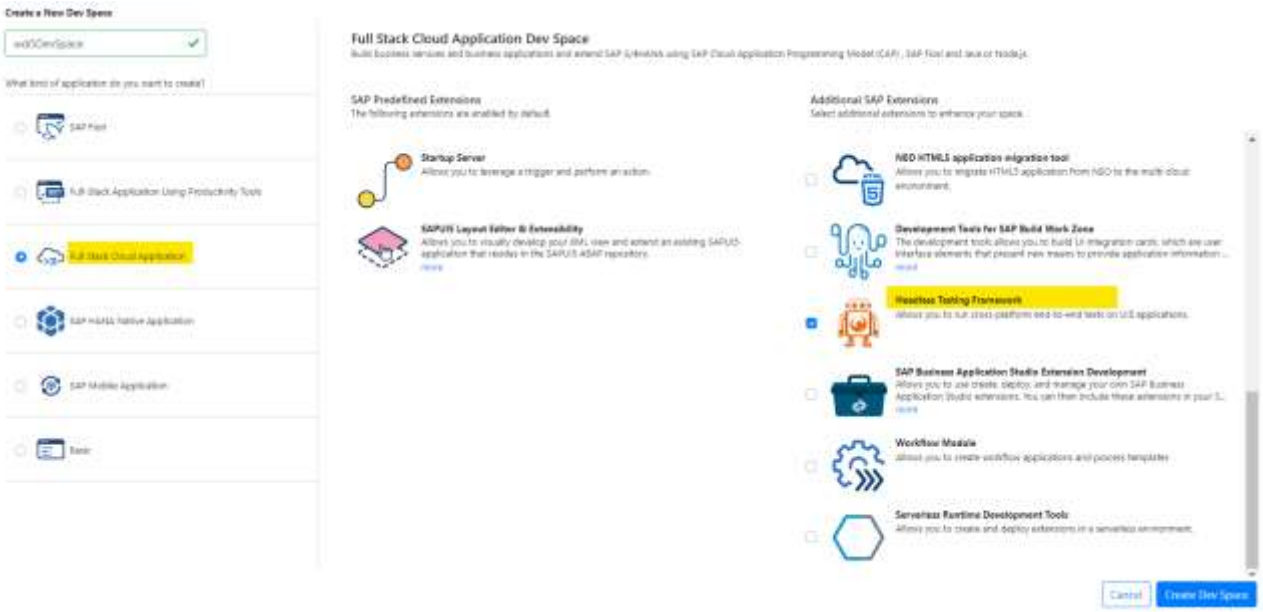
Pre-requisite:

SAP Business Application Studio

If you don’t have one, no worries please create one using the link below.
https://developers.sap.com/tutorials/appstudio-onboarding.html

Step 1: Setting Up BAS

Create a full stack development workspace and enable Headless Testing Framework extension in Dev space:



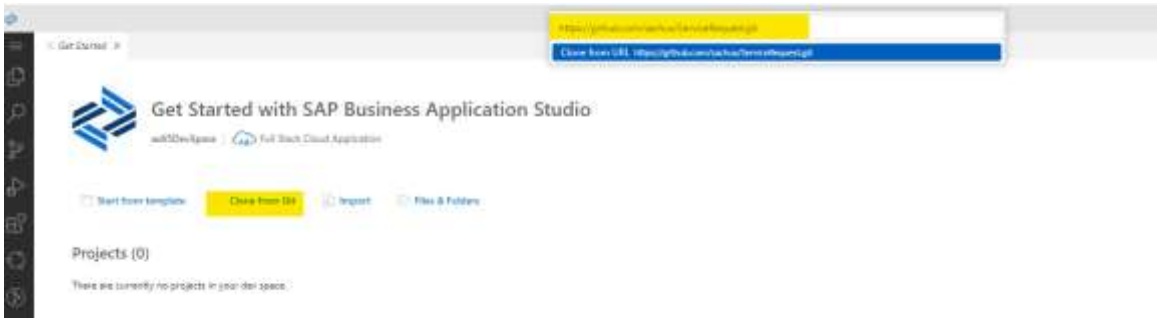
This will install a Firefox driver in the dev space.
Verify that the Firefox driver has been installed correctly using the following commands on the Terminal:

```
# terminal 1: the Firefox version
$> firefox --version
Mozilla Firefox 102.8.0esr
```

```
# terminal 2: the location of the executable
$> which firefox
/extbin/bin/firefox
```

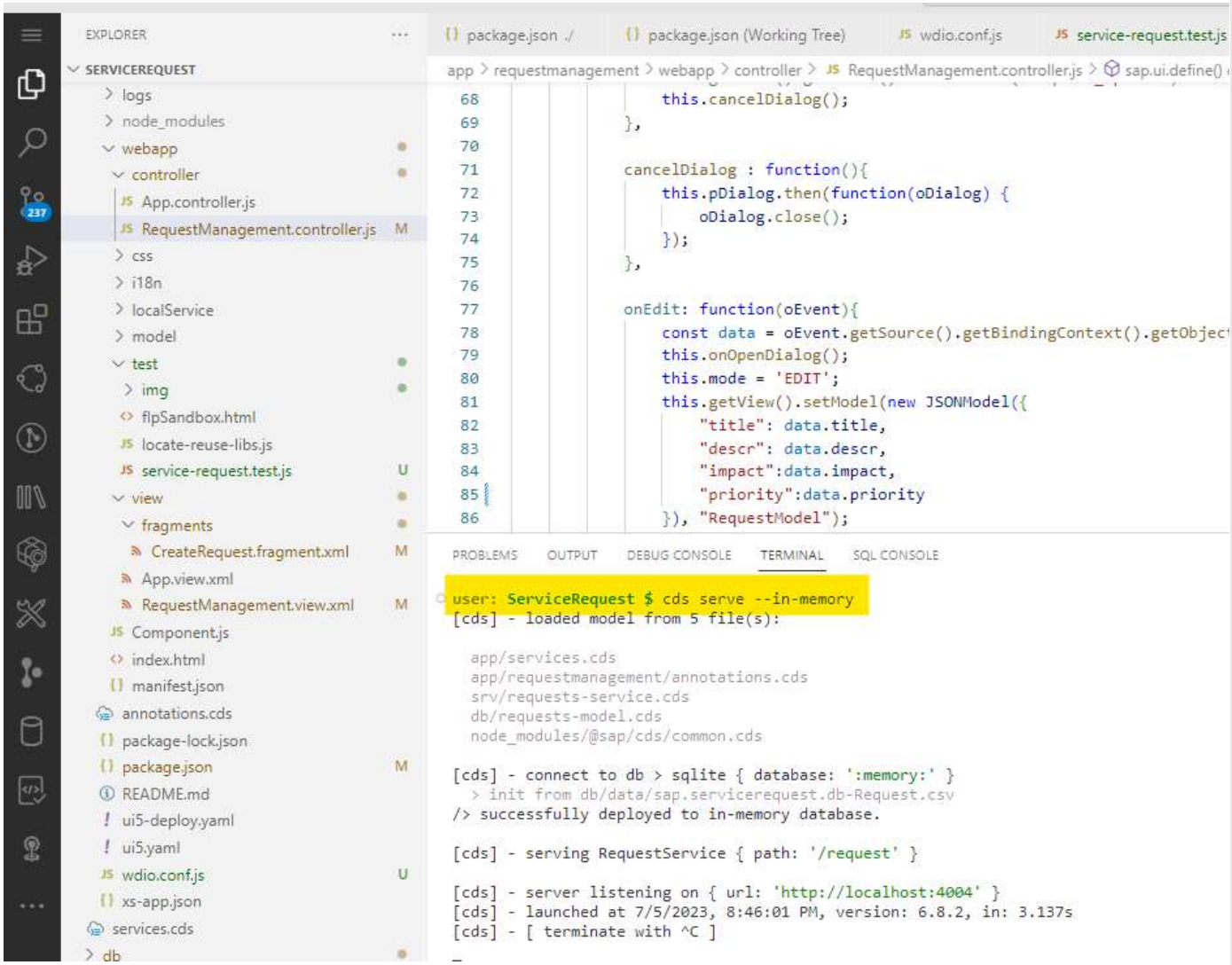
Step 2: Clone Service-Request Application

Clone the UI app code from following repo:
https://github.com/sachux/ServiceRequest.git



Step 3: Run application

Run the app by running command cds serve --in-memory and make sure the application is running.



Step 4: Initialize wdi5

Run the following commands

1. *cd app/requestmanagement*
2. *npm init wdi5@latest*

```
user: ServiceRequest $ cd app/requestmanagement/
user: requestmanagement $ npm init wdi5@latest
Need to install the following packages:
  create-wdi5@0.4.1
Ok to proceed? (y) y
=> copying wdio.conf.js into "./"
 done!
=> installing wdio + wdi5 and adding them as dev dependencies...

up to date, audited 1286 packages in 11s

78 packages are looking for funding
  run `npm fund` for details

38 vulnerabilities (31 moderate, 7 high)

To address issues that do not require attention, run:
  npm audit fix

To address all issues possible (including breaking changes), run:
  npm audit fix --force

Some issues need review, and may require choosing
a different dependency.

Run `npm audit` for details.
 done!
=> adding wdi5 start command ("wdi5") to package.json...
 done!
```

This will:

1. install wdi5 and all required WebdriverIO peer dependencies.
2. add a config file (wdio.conf.js) to your current working directory, using http://localhost:8080/index.html as baseUrl, looking for tests in $ui5-app/webapp/test/**/* that follow the name pattern *.test.js
3. set an npm script named “wdi5” to run wdi5 so you can immediately do npm run wdi5

Step 5: Update Configurations in wdi5

Update the following configurations in *wdio.conf.js*

1. Update wdi5 general settings

```
wdi5: {
       screenshotPath: require("path").join("webapp", "test", "img"), // [optional] {string}, default: ""
       screenshotsDisabled: false, // [optional] {boolean}, default: false;
       logLevel: "error", // [optional] error | verbose | silent, default: "error"
       skipInjectUI5OnStart: false, // [optional] {boolean}, default: false; true when UI5 is not on the start page, you need to later call <wdioUI5service>.injectUI5() manually
       waitForUI5Timeout: 15000 // [optional] {number}, default: 15000; maximum waiting time in milliseconds while checking for UI5 availability
        },
```

2. Replace capabilities with the following code.
   The firefox version and path/to/firefox values appear in the results from the command you ran in the previous step.

```
    capabilities: [
```

```
            {
              acceptInsecureCerts: true,
              browserName: "firefox",
              browserVersion: "102.8.0",
              "moz:firefoxOptions": {
                binary: "/extbin/bin/firefox",
                args: ["-headless"],
                log: { level: "trace" }
              }
            }
          ],
```

3. *Update Base URL:*

```
      baseUrl: "http://localhost:4004/requestmanagement/webapp/index.html",
```

4. Update test runner services

```
      services: ["geckodriver", "ui5"],
```

Step 6: Update package.json

Add following dependencies to the *package.json* in *app/requestmanagement*

```
        "wdio-geckodriver-service": "4.1.1",
        "geckodriver": "^3.2.0",
```

Install dependencies:

-> *npm install*

Step 7: Create a test file

Create a test file called. Service-request.test.js in webapp/tests folder. And have the following code.

```
const { wdi5 } = require("wdio-ui5-service")
const Logger = wdi5.getLogger()


// Describe block for the creation of a request
describe("Creation of request", () => {

  // Async function to get a control based on ID and dialog flag
  const getControl = async function (id, dialog) {

    // Base selector for the control
    const BASE = 'container-com.requestmanagement.requestmanagement---RequestManagement--';
    const selector = {
      timeout: 15000,
      selector: {
        id: `${BASE}${id}`,
        searchOpenDialogs: dialog ? true : false
      }
```

```
    }

    // Return the control obtained from the browser
    return await browser.asControl(selector);
  }

  // Test case to open the dialog
  it("Open Dialog", async () => {

    // Get the "Add New Request" button control
    const createButton = await getControl('addRequest');

    // Get the "text" property of the button and expect it to be "Add New Request"
    const prop = await createButton.getProperty("text");
    expect(prop).toEqual("Add New Request");

    // Take a screenshot of the initial load
    await browser.screenshot('Initial-load');

    // Press the create button to open the request dialog
    await createButton.press();

    // Take a screenshot of the add request popup
    await browser.screenshot('add-request-popup');

    // Get the dialog control for updating the request
    const dialog = await getControl('updateDialog');

    // Get the "title" property of the dialog and expect it to be "Add Request"
    const title = await dialog.getProperty("title");
    expect(title).toEqual("Add Request");
  })
});
```

WebdriverIO and wdi5 can be used with Mocha, Jasmine, and Cucumber, with Mocha being used in all examples in wdi5.
Mocha tests are structured with describe-blocks (“suite”), containing its (“tests”). They can contain hooks, e.g. to run code before all tests (before).

Step 8: Create a folder for Screenshots

*Create a folder at app/requestmanagement/webapp/test/**img***

Step 9: Run the test cases

npm run wdi5

```
user: requestmanagement $ npm run wdi5

> requestmanagement@0.0.1 wdi5
> wdio run ./wdio.conf.js


Execution of 1 workers started at 2023-07-03T22:24:14.483Z

[0-0] RUNNING in firefox - /webapp/test/service-request.test.js
[0-0] PASSED in firefox - /webapp/test/service-request.test.js

 "spec" Reporter:
------------------------------------------------------------------
[firefox 102 linux #0-0] Running: firefox (v102) on linux
[firefox 102 linux #0-0] Session ID: e0a84e66-51be-469f-91ce-c124f7b7e205
[firefox 102 linux #0-0]
[firefox 102 linux #0-0] » /webapp/test/service-request.test.js
[firefox 102 linux #0-0] Creation of request
[firefox 102 linux #0-0]    ✓ Open Dialog
[firefox 102 linux #0-0]
[firefox 102 linux #0-0] 1 passing (3.8s)


Spec Files:      1 passed, 1 total (100% completed) in 00:00:08
```

Step 10: Enhance the test case as given below for Create and delete scenarios

Add the following test cases to the existing test suite

```
 // Test case to test creation of request
  it("Create Request", async () => {

    const table = await getControl('request_Table');
    const countBefore = await ((await table.getBinding("items")).getCount());
    Logger.info(`countBefore:  ${countBefore}`)


    const createButton = await getControl('addRequest');
    createButton.firePress();

    const title = await getControl("title", true);
    const desc = await getControl("desc", true);

    const priority = await getControl("priority", true);
    const impact = await getControl("impact", true);


    const saveButton = await getControl("saveButton", true);

    await title.setValue("There's No Water");
    await desc.setValue("washing machine not filling with water");
    await priority.setSelectedKey("1");
    await impact.setSelectedKey("1");

    await browser.screenshot('Updated Popup');

    await saveButton.press();

    await browser.screenshot('After_Create');
    (await table.getBinding("items")).filter();
    const countAfter = await ((await table.getBinding("items")).getCount());
    Logger.info(`countAfter  ${countAfter}`)
```

```
    expect(parseInt(countBefore) + 1).toEqual(countAfter)

  })


// Test case to test Deletion of request
it("Delete Request", async () => {

    const table = await getControl('request_Table');
    const countBefore = await ((await table.getBinding("items")).getCount());
    console.log("countBefore " + countBefore)

    const deleteLink = await getControl("deleteLink-container-com.requestmanagement.requestmanagement---RequestManagement--request_Table-0", false);
    await deleteLink.press();


    (await table.getBinding("items")).filter();
    const countAfter = await ((await table.getBinding("items")).getCount());
    console.log("countAfter " + countAfter)

    expect(parseInt(countBefore) - 1).toEqual(countAfter)

  })
```

Step 11: Run the test cases

npm run wdi5

```
 "spec" Reporter:
------------------------------------------------------------------
[firefox 102 linux #0-0] Running: firefox (v102) on linux
[firefox 102 linux #0-0] Session ID: 12a3a2db-0035-4bba-89c6-9eae14db1cc0
[firefox 102 linux #0-0]
[firefox 102 linux #0-0] » /webapp/test/service-request.test.js
[firefox 102 linux #0-0] Creation of request
[firefox 102 linux #0-0]    ✓ Open Dialog
[firefox 102 linux #0-0]    ✓ Create Request
[firefox 102 linux #0-0]    ✓ Edit Request
[firefox 102 linux #0-0]    ✓ Delete Request
[firefox 102 linux #0-0]
[firefox 102 linux #0-0] 4 passing (21.6s)

Spec Files:      1 passed, 1 total (100% completed) in 00:00:25
```