**Flexso Intelligent Water Consumption**

SAP Challenge – Hack the Future 2022

AppGyver Track Manual

# Introduction

## Menubar

A picture containing graphical user interface

Description automatically generated

1. UI Canvas: Workspace
2. Launch: Running your app
3. Navigation: Pagination / page structure of your app
4. Data: API & data setup

## Variables



1. View: Workspace / Drag & Drop
2. Variables: predefined variables to be used in your application

# Let’s dive right in!

## Data Source & API Management

1. Go to “Data” in your menubar
   1. Click add data resource
   2. Select REST-API direct integration
      1. Resource ID: FlowChallenge
      2. Short description: /
      3. Resource URL: <https://fb65b981trial-dev-setupfiorifrontend-srv.cfapps.us10.hana.ondemand.com>
2. Go to “Get Collection” in the left side menu.
   1. Relative path: /service/htf2022/FlowStream
   2. Response key path: value
   3. Add a HTTP Header
      1. Label: Autorization
      2. Key: Authorization
      3. Value Type: /
      4. Description: /
      5. Is static: on
      6. Value: Basic SFRGMjAyMkB0ZXN0LmNvbTpwYXNzd29yZA==
   4. Go to Test tab

Graphical user interface

Description automatically generated with medium confidence

* + 1. Click “Run Test”
    2. Click “Set Schema from Response”
  1. Check your schema in the Schema tab.
     1. Your schema should look like this:

Graphical user interface, text, application, email

Description automatically generated

* 1. Remove default properties: ID, createdAt, createdBy, modifiedAt, modifiedBy
     1. This leaves us 3 properties to work with: datetime, descr, flow

## UI Canvas Setup

In this part, we will be dragging items onto our main screen to start building on our application.

Default existing items can be found in the CORE tab, we can upload custom items ourselves and we can install custom items from the global marketplace (and find them at the “Installed”-tab).

Graphical user interface, application

Description automatically generated

Hints:

* Always drag new items underneath existing items
* To reorder the items on the screen, drag & drop them in the item tree

Graphical user interface, text, application

Description automatically generated

Shortcut: Create the exact page layout as the image above.

In detail:

1. Give your page a new name
   1. Click in the gray area around the page.
   2. Under Properties, set your page name

Graphical user interface, application, table

Description automatically generated

1. Go to UI Canvas – View.
   1. Remove the existing items (title & text).
   2. Drag a container onto the screen (CORE – Scroll view).
   3. Drag an Image into the container. (CORE – Image)
   4. Drag a Title into the container (CORE – Image)
   5. Drag a Text onto the screen (NOT in the container). (CORE – Image)
   6. Go to Marketplace

Graphical user interface, application, website

Description automatically generated

* + 1. Search for Google chart &
    2. Install Google charts
    3. Exit Marketplace (upper right corner)
  1. Go to Installed

Graphical user interface, application, Word

Description automatically generated

* 1. Drag a google Chart onto the screen
  2. Drag a new container (CORE – Scroll View)
  3. Drag an Icon List Item onto the screen (CORE - Lists)

Your screen should look like this:

Graphical user interface, text, application

Description automatically generated

Let’s spice things up!

1. Redesign the screen-elements
   1. Image
      1. Download [this](https://github.com/HTF-2022/SetupFioriFrontend/blob/main/app/webapp/img/flexso_icon.ico) image to your computer
      2. Change the extension to .png
      3. Click on the image in appgyver
      4. Set a new source

A picture containing logo

Description automatically generated

* + - 1. Click the image icon on the right (in red)
      2. Upload the recently downloaded Flexso image
      3. Set is as your source (click image & click “Use Image”)
    1. Set new dimensions
       1. Click the image on the screen.
       2. Go to Layout-tab

Graphical user interface, table

Description automatically generated

* + - 1. Under Width and Height, select “Exact size”
         1. Make it 50px Width / 50px Height
  1. Container
     1. Select the container in your Element-tree

Graphical user interface, application

Description automatically generated

* + 1. Go to Layout-Tab
       1. Under Layout:

Graphical user interface

Description automatically generated

Voila, looks a lot better now!

Let’s add some text next.

1. Texts
   1. Select your title-element
      1. Under Properties, change the content to “Flexso Flow Meter”
   2. Select your text-element
      1. Under Properties, change the content to “Interactive dashboard”.

Looking good!

## Google Charts

Let’s move on to the more difficult stuff. Creating a real-time graph perhaps?

For this, we will need some “Variables”. There are multiple types of variables.

There are Data-variables, they are used to store data from our API’s.

There are App-variables, they are used to store values to be used throughout the whole application (look at them as “global variables”).

Lastly, there are Page-variables. They can be used on all pages, but the values will be emptied when you change pages. These are recommended since they take less memory (by being cleared), and they will be reusable throughout different pages.

For every variable we want to use, we need to give it a typedefinition.

Let’s start by creating our Data-variable to store our API-data, and create our page-variables for the graph after.

Variables can be created in the Variables-view. Enter it by clicking “Variables”.



Important: Variable names are case sensitive! So, pay close attention.

1. Adding variables
   1. Go to Data-variables
      1. Click “Add data variable”
      2. Select “FlowChallenge” -> This is the schema you’ve set when creating the API, remember? (2.1.2.e)
   2. Go to Page-variables
      1. Create a variable called “chartData”
         1. Variable value type: List (Complex Type)
            1. List item type: List

List item type: Any value (Complex Type)

* + 1. Create a variable called “chartFunctionString”
       1. Variable value type: Text
    2. Create a variable called “listHeader”
       1. Variable value type: List (Complex Type)
          1. List item type: List

List item type: Any value (Complex Type)

* + 1. Create a variable called “listItem”
       1. Variable value type: List (Complex Type)
          1. List item type: List

List item type: Any value (Complex Type)

This should leave you with these variables:

A picture containing application

Description automatically generated

Alright. Now we’re set up with our variables, we can start creating a graph!

Go back to the Components view.



1. Click in the gray area around your screen (this will select the page itself)
   1. At the bottom of your screen, click “Add logic to Home”

In this view, we will be setting values to our variables and add dynamic flows to our app.

We will be setting our graph header, graph data and the graph itself. We will also make the graph auto-updating itself with data, so we have real-time data!

I you need a cup of coffee... Now is the right time!

1. Under Logic – Core
   1. Under Data, select “Get Record Collection”
      1. Drag it behind “Page Mounted”
      2. Connect “Page Mounted” with “Get Record Collection” by dragging a line between the dots.
      3. Rename the item to “Get Flows”
         1. Hint: Under Properties – Advanced: Name
   2. Under Variables, select “Set data variable”
      1. Drag it behind “Get Flows”.
      2. Connect “Get Flows” with “Set data variable” (use the top dot, this means the call wass successful, the bottom dot means the call has failed).
         1. Under Properties, click the Array icon
            1. Select Output value of another node
            2. Select logic node: “Get Flows”
            3. Select node output: Collection of records
   3. Under Variables, select “Set page variable”
      1. Drag it behind “Set data variable”
      2. Connect “Set page variable” with “Set data variable”
         1. Under Properties, click on the Variable name and select listHeader
         2. Assigned value: “[['datetime', 'flow']]”
      3. Rename the item to “Set ListHeader”
   4. Under Variables, select “Set page variable”
      1. Drag it behind “Set ListHeader”
      2. Connect “Set page variable” with “Set ListHeader”
         1. Under Properties, click on the Variable name and select listItem
         2. Assigned value:
            1. Select Formula
            2. Formula: “MAP<item>(data.FlowChallenge1, [LOOKUP(item, "datetime"), NUMBER(LOOKUP(item, "flow"))])”
      3. Rename the item to “Set ListItems”
   5. Under Variables, select “Set page variable”
      1. Drag it behind “Set ListItems”
      2. Connect “Set page variable” with “Set ListItems”
         1. Under Properties, click on the Variable name and select chartData
         2. Assigned value:
            1. Select Formula
            2. Formula: “UNION(pageVars.listHeader, SLICE(pageVars.listItem, COUNT(pageVars.listItem)-10, COUNT(pageVars.listItem)))”
      3. Rename the item to “Set ChartData”

Alright. We’re getting close!

We’ve now set up all necessary data to use in our graph.

Let’s create the graph next!

1. Under Logic – Core
   1. Under Advanced, select “Javascript”
      1. Drag it behind “Set ChartData”
      2. Double click to open.
      3. Copy paste this code: (double click the code to open)



* + 1. Change the output:
       1. Value type: Object
       2. Remove the existing property “result”
       3. Add new property “funcAsString”
       4. Property Type: Text
       5. Be sure that the property is required (checkbox)
    2. Try to add a legend or descriptions for the axis, and a title to the graph. You can find the documentation [here](https://developers.google.com/chart/interactive/docs/gallery/linechart).
    3. Click Update
    4. Rename the Javascript-element to “Generate Google LineChart”
  1. Under Variables, select “Set page variable” (last time, I swear)
     1. Drag it behind “Generate Google LineChart”.
     2. Connect “Generate Google LineChart” with “Set page variable”
        1. Under Properties, click on the Variable name and select “chartFunctionString”
        2. Assigned value:
           1. Select “Output from another node”
           2. Select “Generate Google LineChart”
           3. Select “funcAsString”
     3. Rename the element to “Set ChartString”.

Alright. We’re almost there. Final steps…

1. In the UI Canvas screen
   1. Select the Google Graph
      1. Set Chart function: Data & Variables: Page Variables: chartFunctionString
      2. Set Chart data: Data & Variables: Page Variables: chartData

Finally… Done!

Or are we?

We wanted a real-time graph... We’re not there yet, but let’s look at our product so far.

We’ve come a long way!

## Running the application

To run our application, the easiest way is to use our phone and install the AppGyver application from the Appstore.

When we sign in, we will be able to select our project and open our application.

It’s a miracle! Our first graph is right there!

If it isn’t… Well, double check or ask the Flexso team to take a quick look 😊.

## Real time graph

When everything works, well be going for the final thing: We want a real-time graph. We won’t be showing you the full solution, there’s some work left for you to finalize it!

1. Go back to the logic screen.
   1. Under Logic – Core – Utility, select “Delay”
   2. Drag it behind “Set ChartString”
   3. Connect “Delay” input node with the “Get Flows” output node.
      1. This will start the delay when you receive data.
   4. Set the delay for 2 seconds.
      1. Click Delay
      2. Set Time to wait: 2000
      3. Set unit: ms
   5. Connect “Delay” output dot with the input dot of “Get Flows”
      1. This will call the “Get Flows”-API again!

We now created a loop te retrieve the API‘s data every 2 seconds.

But did you see that after “Get Flows”, not only the delay is called, but also the whole graph flow?

This means that every 2 seconds, we will retrieve data, set all our page variables and regenerate our chartdata!

Let’s check this out on our phone!

Do you notice something?

…

Not only is the data regenerating, but also the whole chart regenerates visually…

That’s not what we want…

Can YOU find a solution for this?

Let’s see if you masted SAP AppGyver already!

## Extensions

You might be wondering what the list element on the screen is for.

This allows you to have multiple pages to navigate to.

Example:

1. Select the Icon list item element
   1. Click the “Repeat with” icon
      1. Select Formula
      2. Formula: “[{"Name": "Last 24 hours"},{"Name": "This week"},{"Name": "This month"}]”
         1. This is just an example. Every item in this array can be connected to a new page.
   2. Set Primary label
      1. Select Data item in repeat
         1. This will use the list you just provided
      2. Select repeat: current
      3. Select repeat data property: current – Name

Now check your phone!

Looks great, isn’t it!

Now, try to create a page.

1. Go to the page overview.

Graphical user interface, text

Description automatically generated

* 1. You’ll see the existing “Home”-page.
  2. Add a new page and call it “This month”

1. Now go to the Navigation tab in the menu bar
   1. Add a new item
   2. Under Properties, change the Page to “This month”
      1. You will now be able to navigate to the new page
   3. To make it easier to understande, change the labels of both pages to “Home” and “This month” correspondently
2. Go to the UI Canvas of the Home page
   1. Select the icon list item element
   2. Go to the logic screen “Add logic for icon list item 1”
      1. From Logic – Core – Utility, drag an “If condition” next to “component tap”
      2. Connect them
      3. Select the if condition
         1. At Properties, click the icon at Condition
            1. Select Formula
            2. Formula: repeated.current.Name == "This month"
            3. Click save
      4. From Logic – Core – Navigation, drag an “Open Page” next to the if condition.
         1. Connect the top node of the “If condition” element with the “Open page” element
      5. Select the open page element
         1. Under Properties – Page, select “This Month”.

Now test it on your phone!

To add new pages, just add combinations of new if conditions and open page elements and connect them to the bottom node of the previous if condition.

Example:

* New page: This week
  + - 1. Create a new page “This week” in the page overview
      2. Create new navigation in the navigation menu
      3. Add if condition (with correct formula) and an open page element leading to the correct page.

Like this:

Graphical user interface, diagram

Description automatically generated

Now, you have all the tools to extend this application and to do some creative stuff with it.

Add new pages with new graphs.

Do new calculations by adding extra variables… Reuse the code, elements and knowledge you already have and make something of your own.

[AppGyver](https://docs.appgyver.com/) has great documentation, aswell as [Google Charts](https://developers.google.com/chart/interactive/docs/gallery/linechart). So don’t let issues stop you and Google your way into solutions!

Add pictures, reuse the other API calls like the GandalfQuotes, do some crazy stuff and make the application something you would like to use yourself!

Everything is allowed, even sounds, gifs, videos, images…

If you have any questions at any point, ask your teamleaders for a hint.

You will get points for interest and motivation, and **you won’t lose points for asking questions**!

Give it your best!

Good luck!

The Hack The Future Team.