

# Building a Mobile App with Oracle Visual Builder Cloud Service

In this lab you'll create an application that automates a process of approving travel requests that is currently done in spreadsheets. You will also call a public REST service and incorporate the results into your application.

## Connect to Oracle Visual Builder Cloud Service (VBCS)

Open the chrome browser and navigate to:

<https://oicvbcs-gse00015502.uscom-east-1.oraclecloud.com/ic/builder/>

Login: cloud.admin

Password: Stoic@1LUster

## Creating a New Application

First we will create an empty application.

1. Click the **New** button.
2. Name your application **<YourName>-Travel Approval**.

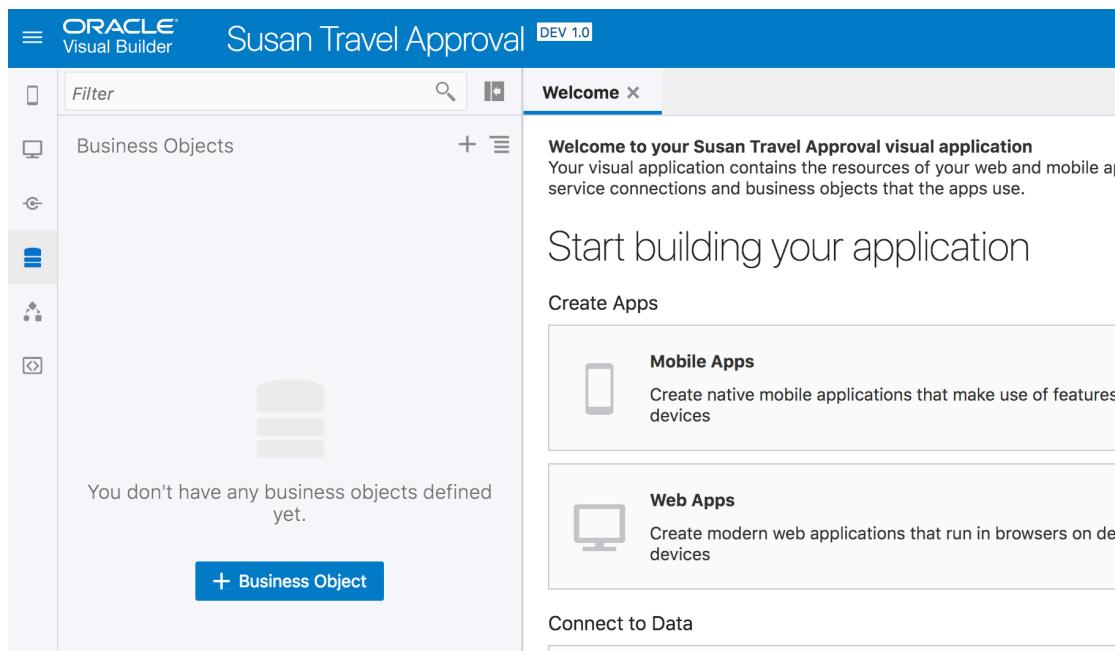


3. Click **Finish** to complete the creation of your application.

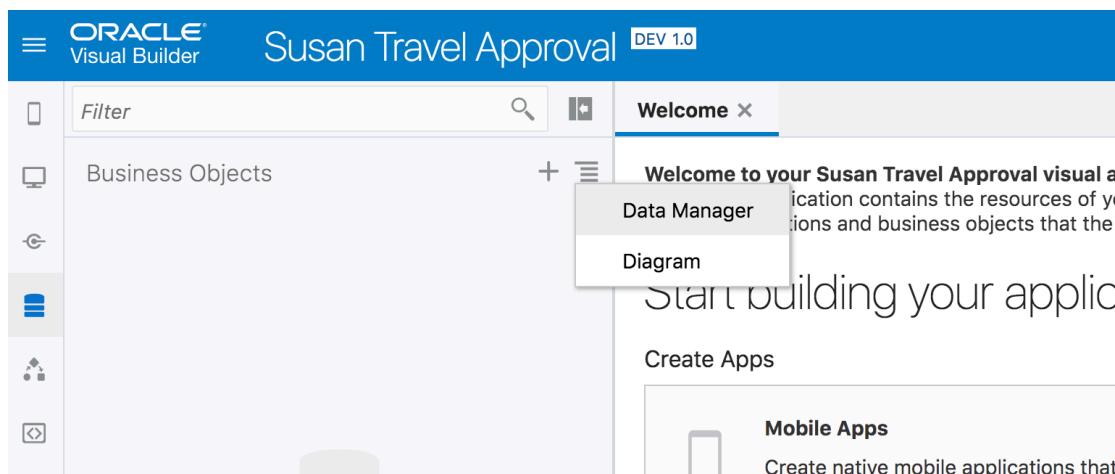
## Creating Business Objects

You will create Business Objects to store data by importing an existing spreadsheet. You will then edit the Business Objects, adding fields and setting default values and creating a relationship between objects

1. Click on **Business Objects icon** to the left of the navigator. Note that the **Business Objects** navigator opens in the left panel.



- At the top of the navigator, click the **Business Objects** menu and select **Data Manager**. This provides short cuts when creating business objects.



- Click **Import Business Objects** to import data from a spreadsheet. Note click the purple Import Business Objects, not the orange **Import From File**.

The screenshot shows the Oracle Visual Builder Data Manager interface. At the top, there's a navigation bar with icons for Filter, Welcome, and Data Manager (which is currently selected). Below the navigation bar, there's a sidebar on the left with icons for Business Objects, Data Sources, and other system components. The main area displays a message: "You don't have any business objects defined yet." Below this message is a blue button labeled "+ Business Object". To the right of this area is a large purple button with an upward arrow icon, labeled "Import Business Objects". Above the purple button, there's a section titled "Import from File" with the sub-instruction: "Replace all data in Development with the data from a zip file, CSV file or Excel spreadsheet." On the far right, there are "Export" and "Development" buttons.

4. Inside the Import Business Objects wizard upload the [Flights](#) spreadsheet.

The screenshot shows the "Import New Business Objects" wizard. The title bar says "Import New Business Objects". Below it, there are three numbered steps: 1. Upload File, 2. Business Objects, and 3. Fields. Step 1 is active, indicated by a blue circle with the number 1. A text input field is labeled "Provide the file containing data that you wish to import." Below the input field is a dashed box where a file named "Flights.xlsx" is shown with an upward arrow icon. At the bottom of the screen, there's a log message: "Uploading Flights.xlsx... Travel Requests Business Object... found 8 exist Travel Requests records... found 8 exist Airline Business Object... found 5 exist Airline records... found 5 exist Upload succeeded."

5. Once you see the upload has succeeded click **Next**.
6. Change the Name After Import and New Object ID from **TravelRequests** to **TravelRequest** and from **Airlines** to **Airline**.

Import New Business Objects

Cancel < Back

1      2      3

Upload File      Business Objects      Fields

**Business Objects**

We've found these candidate Business Objects in the import file which can be imported as new Business Objects. You can also change the singular Business Objects.

Filename	Name After Import	New Object ID
Flights.xlsx / Airlines	Airline	Airline
Flights.xlsx / Travel Requests	Travel Request	TravelRequest

7. Click **Next**.
8. Click **TravelRequest** to see details of that Business Object.
9. Change the **Type** of the field with the display label **Date** field from **DateTime** to **Date**.

Import New Business Objects

Cancel < Back

1      2      3

Upload File      Business Objects      Fields

**Fields**

Select a Business Object to see its fields. You can change the field names and data types.

Datatype Conversion Options ? Fail on Exception ▾

Airline    **Travel Request**

ID	Display Label	Type	Required	Sample
cost	Cost	#	<input type="checkbox"/>	1200
date1	Date	datetime	<input type="checkbox"/>	2018-02-06T00:00:00
from1	From	Date	<input checked="" type="checkbox"/>	
name	Name	A	<input type="checkbox"/>	
picture	Picture	#	<input type="checkbox"/>	
to1	To	#	<input type="checkbox"/>	

10. Click **Finish** to complete the data import and **Close** to finish. **Note:** If your business objects do not appear in the list of business objects, refresh your browser.

Visual Builder creates the business objects – it creates a table in the database, adds primary key (ID) and audit fields, imports the data from Excel into it and exposes a set of REST services that allow you to do the CRUD on that new business object

## Editing the Business Objects

Now you will start adding additional fields to the business objects. We'll start by adding a Boolean field to track whether the request is approved.

1. In the **Business Object navigator**, click on **Travel Request** to open it in the editor

2. Click on the **Fields** tab
3. Click on **New Field** to add a new field to this business object

The screenshot shows the Data Manager interface with the 'Travel Request' business object selected. The 'Fields' tab is active. A modal window titled '+ New Field' is open, showing the configuration for a new field named 'approved'. The 'Type' is set to 'Boolean'. The 'Label' is 'Approved'. The 'Id' is 'approved'. The 'Display Label' is 'Approved'. The 'Required' checkbox is checked. The 'Description' is empty.

Type	Id	Display Label	Required	Description
#	id	Id	✓	
⌚	creationDate	Creation Date	—	
⌚	lastUpdateDate	Last Update Date	—	
A	createdBy	Created By	—	
A	lastUpdatedBy	Last Updated By	—	
A	name	Name	—	
A	from1	From	—	
⌚	date1	Date	—	
⌚	approved	Approved	—	

4. Add a field called **Approved** of type **Boolean**

The screenshot shows the Data Manager interface with the 'Travel Request' business object selected. The 'Fields' tab is active. A modal window titled '+ New Field' is open, showing the configuration for a new field named 'approved'. The 'Type' is set to 'Boolean'. The 'Label' is 'Approved'. The 'Id' is 'approved'. The 'Display Label' is 'Approved'. The 'Required' checkbox is checked. The 'Description' is empty.

Type	Id	Display Label	Required	Description
#	id	Id	✓	
⌚	creationDate	Creation Date	—	
⌚	lastUpdateDate	Last Update Date	—	
A	createdBy	Created By	—	
A	lastUpdatedBy	Last Updated By	—	
A	name	Name	—	
A	from1	From	—	
⌚	date1	Date	—	
⌚	approved	Approved	✓	

5. Select the **Approved** field. In the **Property Inspector** on the right, scroll down to the **Value Calculation** section. Select **Set to default if value not present** and set the default value to **false**.

The screenshot shows the Data Manager interface with the 'Travel Requests' business object selected. The 'Fields' tab is active. A modal window titled '+ New Field' is open, showing the configuration for a new field named 'approved'. The 'Type' is set to 'Boolean'. The 'Label' is 'Approved'. The 'Id' is 'approved'. The 'Display Label' is 'Approved'. The 'Required' checkbox is checked. The 'Description' is empty.

**Property Inspector (right side):**

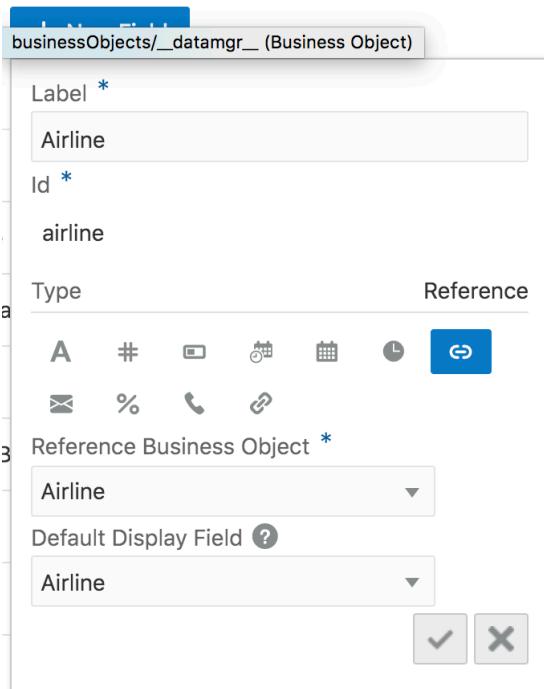
- Value Calculation:**
  - None
  - Set to default if value not provided
  - false
- Calculate value with a formula
- Aggregate from related object data
- Constraints:**

Type	Id	Display Label	Required	Description
#	id	Id	✓	
⌚	creationDate	Creation Date	—	
⌚	lastUpdateDate	Last Update Date	—	
A	createdBy	Created By	—	
A	lastUpdatedBy	Last Updated By	—	
A	traveler	Traveler	—	
A	from1	From	—	
A	to1	To	—	
#	cost	Cost	—	
⌚	date1	Date	—	
⌚	approved	Approved	—	

## Adding a Relationship to Another Object

You will now create a reference from the **Travel Request** object to the **Airline** object.

1. Add another field to **Travel Request**. Set the name to **Airline** and the type to **Reference**.
2. Select **Airline** as the **Business Object**. The tool automatically uses the **ID** field as the foreign key. Select the **Airline** field as the **Default Display Field**.



3. Click on the Endpoints tab. Note the REST Endpoints that have been exposed to access the Business Objects in your apps

Endpoints			
Filter Endpoints			
<b>TravelRequest</b> 5 endpoints (0 child objects)			
<b>GET</b>	/TravelRequest	getall_TravelRequest	Get Many
<b>POST</b>	/TravelRequest	create_TravelRequest	Create
<b>GET</b>	/TravelRequest/{TravelRequest_Id}	get_TravelRequest	Get One
<b>PATCH</b>	/TravelRequest/{TravelRequest_Id}	update_TravelRequest	Update
<b>DELETE</b>	/TravelRequest/{TravelRequest_Id}	delete_TravelRequest	Delete
<b>TravelRequest/airlineObject</b> 5 endpoints (0 child objects)			
<b>GET</b>	/TravelRequest/{TravelRequest_Id}/child/...	getall_TravelRequest-airlineObject	Get Many
<b>POST</b>	/TravelRequest/{TravelRequest_Id}/child/...	create_TravelRequest-airlineObject	Create
<b>GET</b>	/TravelRequest/{TravelRequest_Id}/child/...	get_TravelRequest-airlineObject	Get One
<b>PATCH</b>	/TravelRequest/{TravelRequest_Id}/child/...	update_TravelRequest-airlineObject	Update
<b>DELETE</b>	/TravelRequest/{TravelRequest_Id}/child/...	delete_TravelRequest-airlineObject	Delete

## Adding Aggregate Fields to an Object

Now you will add some fields that aggregate data in the Airline object. We will use this data to populate charts in the web apps showing statistics of our trips.

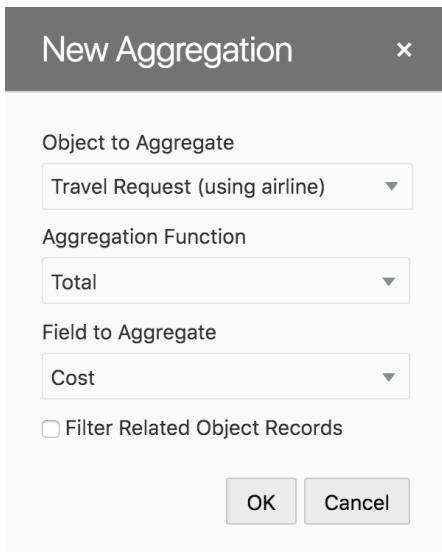
1. Select the **Airline** object's editor by clicking **Airline** in the list of business objects.
2. Add a field: **Total Cost**, of type **Number** that is going to show the total cost of air travel expense requests by airline.

The screenshot shows the 'Fields' tab of the 'Airline' object's editor. A modal window titled '+ New Field' is open, showing the configuration for a new field named 'Total Cost'. The field is of type 'Number'. The 'Value Calculation' section is visible, with the 'Aggregate from related object data' option selected. Other options like 'None', 'Set to default if value not provided', and 'Calculate value with a formula' are also shown.

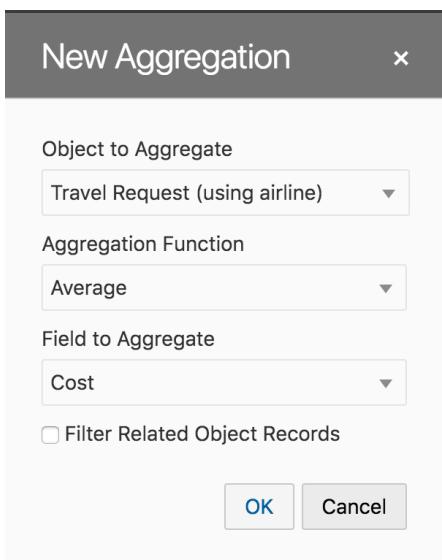
3. With the **Total Cost** field selected, go to the Value Calculation section of the Property Inspector and select **Aggregate from related object data**

The screenshot shows the 'Value Calculation' section of the Property Inspector for the 'totalCost' field. The 'Aggregate from related object data' option is selected. Other options like 'None', 'Set to default if value not provided', and 'Calculate value with a formula' are also listed.

4. Click the + **Edit Aggregation** button. See that the object to be aggregated has defaulted to Travel Request, using airline. Select **Total** as the function and select **Cost** as the field to be totaled. Click OK



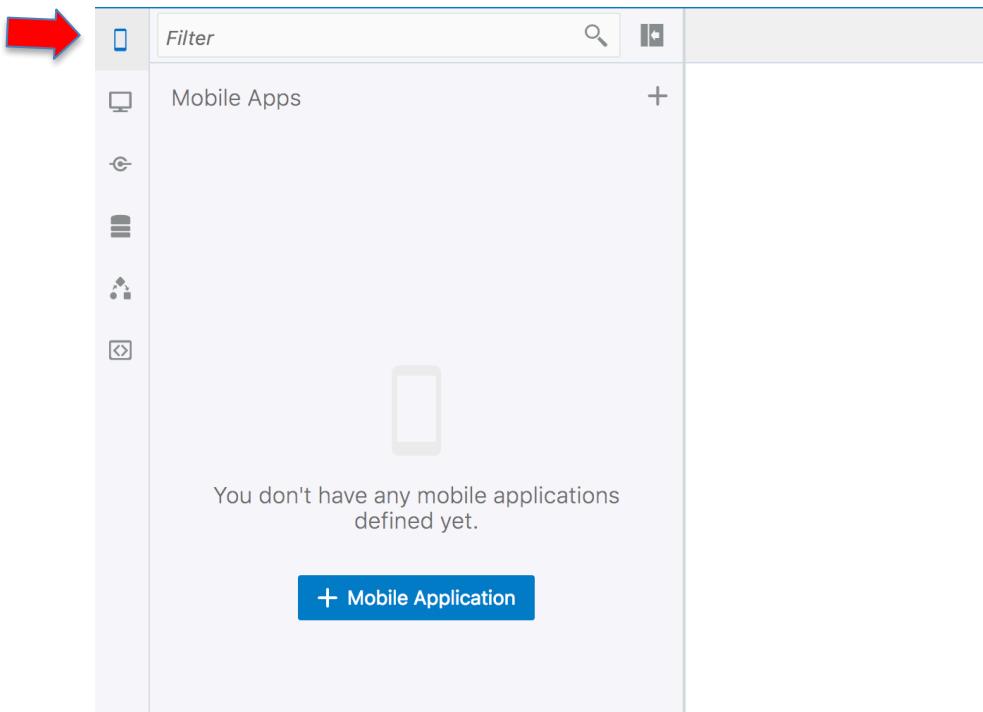
5. Add a field: **Average Cost**, type **Number**, that is going to show the average cost of air travel expense requests by airline
6. Again, with the **Average Cost** field selected, go to the Value Calculation section of the Property Inspector and select **Aggregate from related object data**
7. From the **Travel Request** object, select **Average** as the function and **Cost** as the field to aggregate



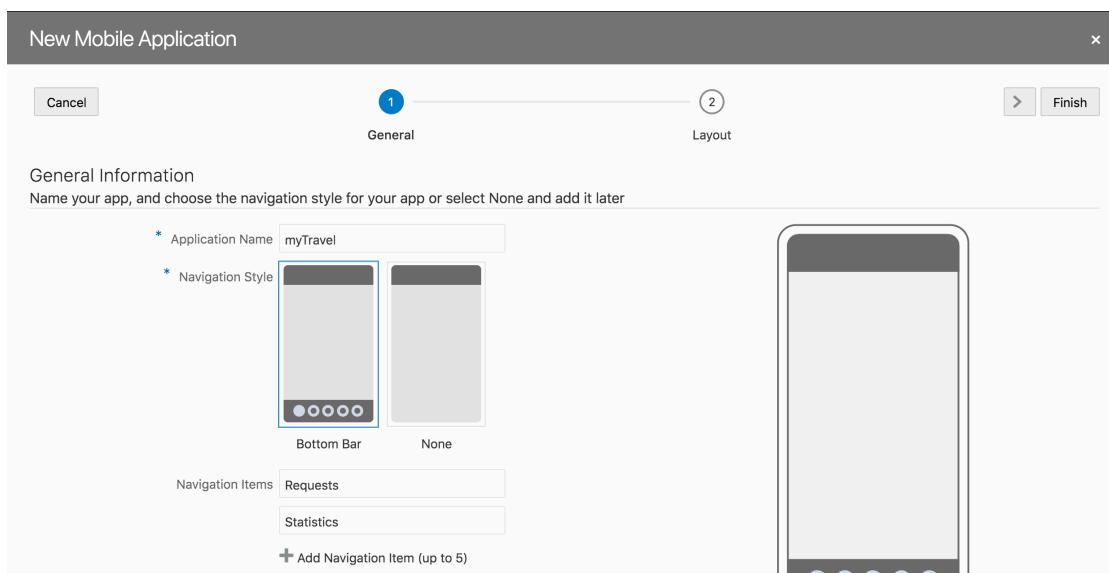
## Create the Mobile Application

In this section you will create a mobile app to enable the travel approval process, using the Business Objects you created earlier

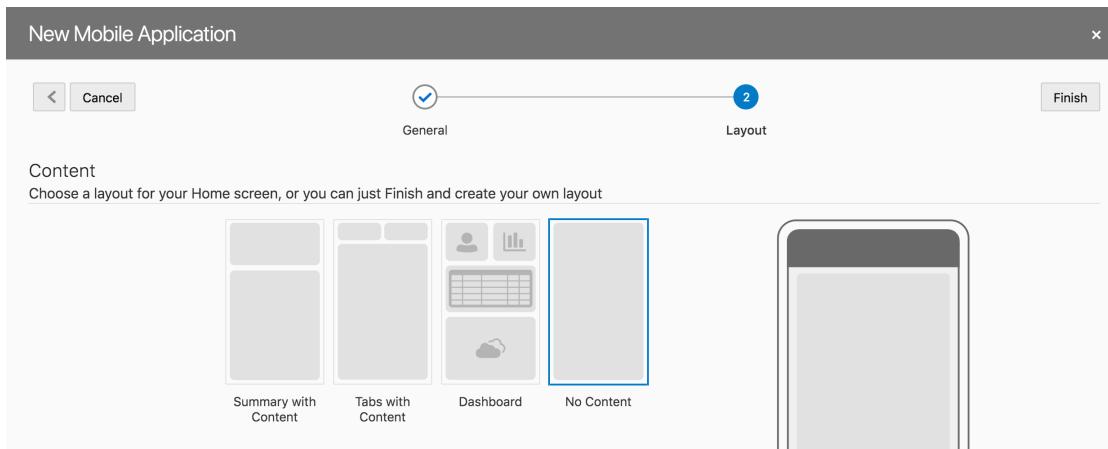
1. Click the **Mobile Apps** icon in the navigator and click the **+ Mobile Application** button (you can also close the open tabs in the editor)



2. Name the Mobile Application **myTravel**. Keep the default **Navigation Style (Bottom Bar)**.
3. Name the first two **Navigation Items Requests** and **Statistics**. Delete the third navigation item.



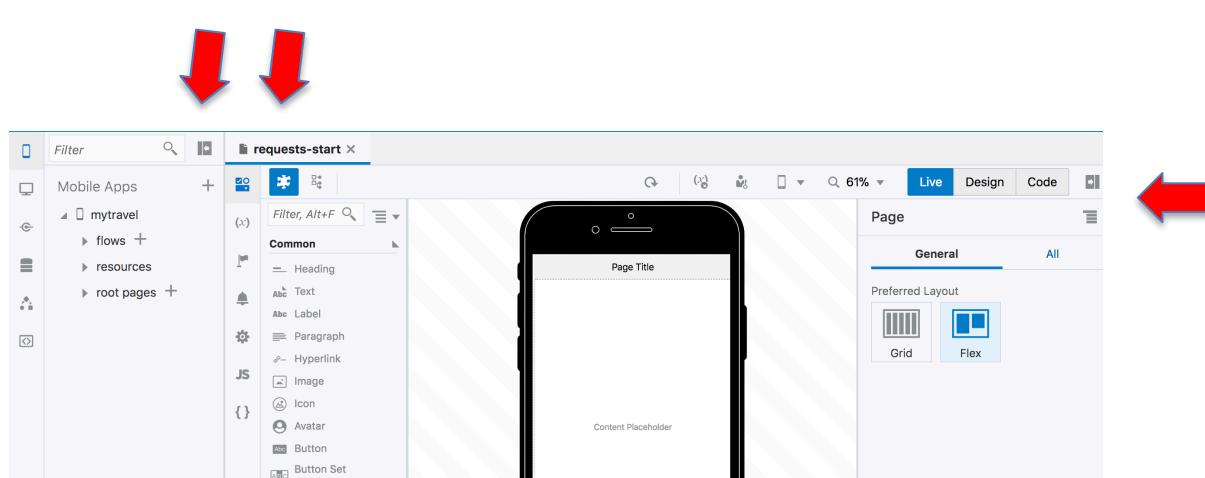
4. Click the > button. Now you can choose a default layout for your home screen. Select **No Content**.



## 5. Click Finish.

VBCS creates the application. Expand it in the **Mobile Apps** window to see its structure. The app has a page flow for each of the **Navigation Items** you specified earlier. Each flow has a default home screen and the first of these, **requests-start**, is open in the **Editor**. This is the entry point to your application.

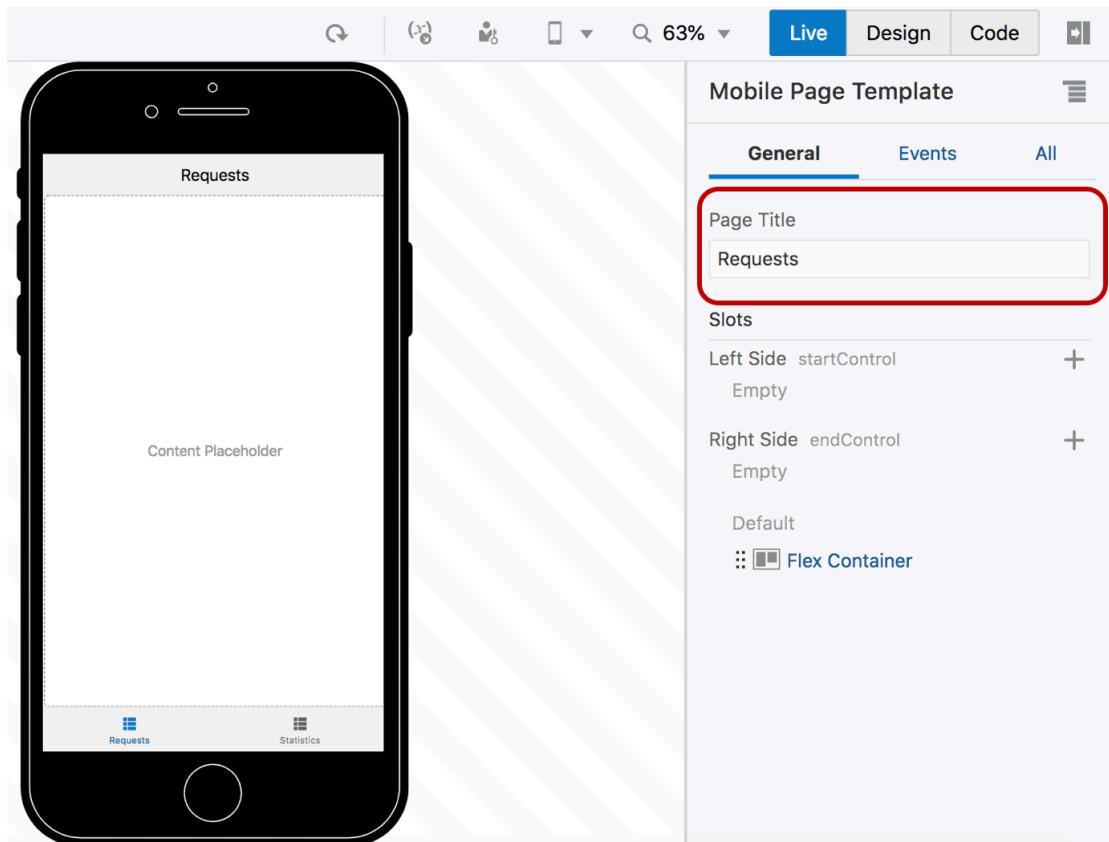
You can collapse the **Application Navigator** and the **Page Structure** viewer as necessary by clicking on the collapse icons.



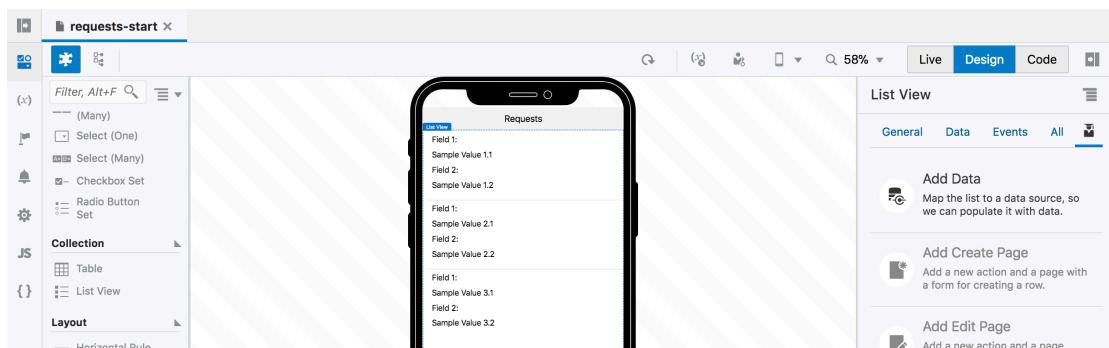
Now you are in the **Visual Page Editor**. On the left is the **Component Palette**, the main area is the **Page Editor** and to the right is the **Property Inspector**.

## Adding a List to the Screen

1. In the **Editor**, select **Page Title**. The **Property Inspector** updates to show the properties of the Mobile Page Template, which includes the text **Page Title**.
2. Change the **Page Title** to **Requests**.



### 3. From the Component Palette drag a **List View** onto the page



## Binding Data to the Screen

In this section you bind the elements on your screen to data – in this case the **TravelRequest** business object. You access and update the data in the underlying table using the REST Endpoints that were created as part of the business object when you imported the spreadsheet.

We will use **Quick Starts** to quickly bind our table and generate our Edit page for the data.

1. In the **Property Inspector**, note the **Quick Start** tab is open by default when you first drop a List View on the page.

The screenshot shows the 'List View' interface. At the top, there are tabs: 'General', 'Data', 'Events', 'All', and a fourth tab which is highlighted with a red box and has a blue underline. Below these tabs, there are three options: 'Add Data' (with a clipboard icon), 'Add Create Page' (with a document icon), and 'Add Edit Page' (with a document icon). The 'Add Data' option is the one highlighted.

2. Click **Add Data**.
3. Select the **TravelRequest** Business Object and click **Next**.

The screenshot shows the 'Add Data' wizard. The title bar says 'Add Data'. The progress bar at the top indicates Step 1 ('Locate Data') is active. The main area is titled 'Choose an Endpoint' with the sub-instruction 'Select a data source for your list.' Below this, there's a section 'Choose the source of your data' with a 'Filter' button. Under 'Business Objects', there are three icons: 'Airline' (gray), 'TravelRequest' (blue, highlighted with a red box), and 'TravelReques...' (gray). A 'Locate Data' button is located above the business objects section.

4. Now choose the **List Item** template and click **Next**.

The screenshot shows the 'Add Data' wizard. The title bar says 'Add Data'. The progress bar indicates Step 2 ('Select Template') is active. The main area is titled 'Select List Item Template' and shows two options: a grid-based template with numbered slots (1 through 5) and a more flexible template with numbered slots (1 through 5) and a note: 'Templates are flexible. You can put multiple fields into a single spot. You can leave any of the spots empty - the template will adapt.' Below the templates, there are 'Locate Data', 'Select Template', 'Bind Data', and 'Define Query' buttons.

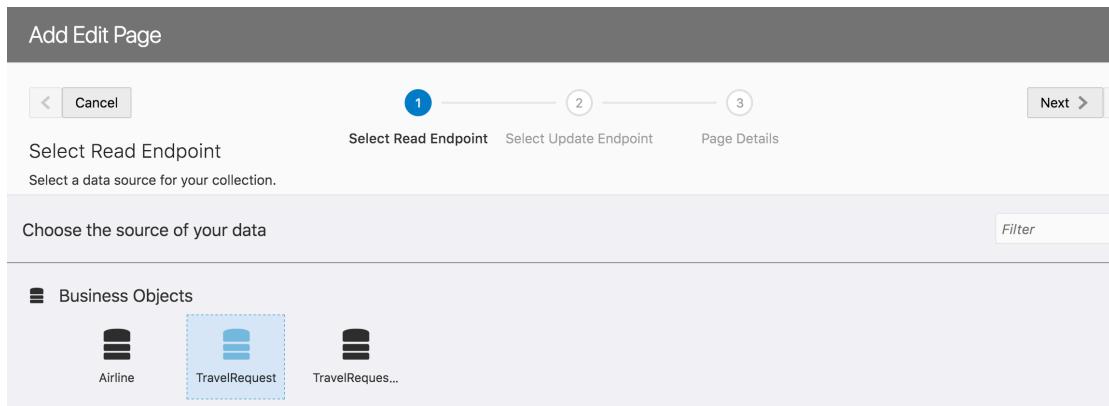
5. Now select which fields to show in the list. Drag and drop the following fields to the slots in the wizard as shown in the picture below:

  - **picture**
  - **name**
  - **to1**
  - **cost**
  - **airlineObject > items > airline**

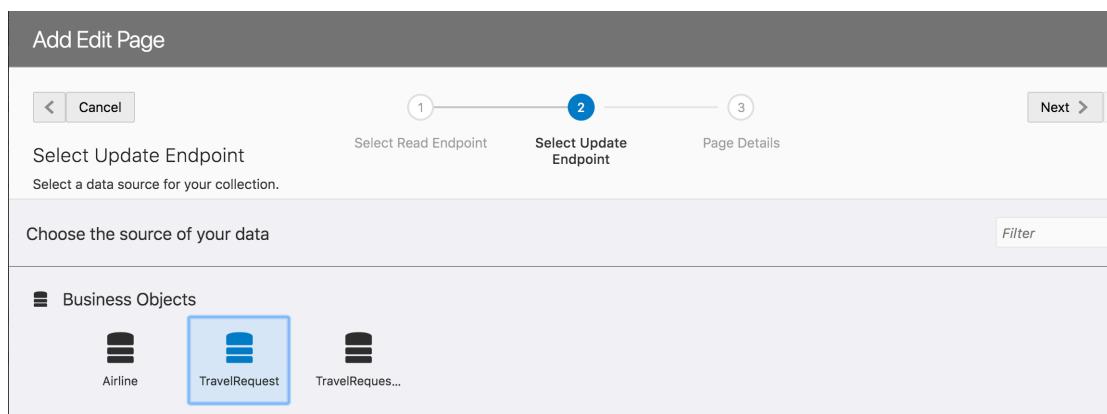
6. Click **Next** and then **Finish**
7. Note that the Editor is now showing the live data from the Business Object, even in Design view

## Adding an Edit Page

1. In the Quick Start panel of the Property Inspector, click **Add Edit Page**.
2. Select the **TravelRequest** Business Object as the source data for this edit page and click **Next**.



3. In the **Update Endpoint** page of the wizard, select the **TravelRequest** business object and click **Next**.



4. Select the following fields to be displayed on the Edit Page:

- **picture**
- **name**
- **approved**
- **date1**
- **cost**
- **airline**
- **to1**

You can reorder the fields once you've selected them using drag and drop. Note that you select the **airline** reference field, not the **airlineObject**.

**Page Details**  
Specify the Edit Page details.

**Update Endpoint Structure**

- request
  - ✓  airline
  - airlineObject
  - ✓  approved
  - ✓  # cost
  - A createdBy
  - creationDate
  - ✓  date1
  - A from1
  - # id
  - lastUpdateDate
  - A lastUpdatedBy
  - ✓  A name

**Fields**

A picture	X
A name	X
approved	X
date1	X
# cost	X
# airline	X
A to1	X

**Button label**

Edit TravelRequest
Page title
Edit TravelRequest
Page name
requests-edit-travel-request

5. Click Finish.

## Adding Navigation from the List View to the Edit Page

Now you set up the selection event on the **Requests** list to call an action chain that navigates you to the corresponding Edit page.

1. Make sure the **List View** component is selected. You can select it in the **Page Editor** or in the **Page Structure** window.

**Page Structure**

- Mobile Page Template
  - Flex Container (List View)
    - List View
    - contextMenu
    - itemTemplate
      - Row Template
        - List Item
        - Image
        - Image

**List View**

General Data Events All

Selection Mode: none

2. On the General Properties for the List View – set the **Selection Mode** property to **Single**.

**Page Structure**

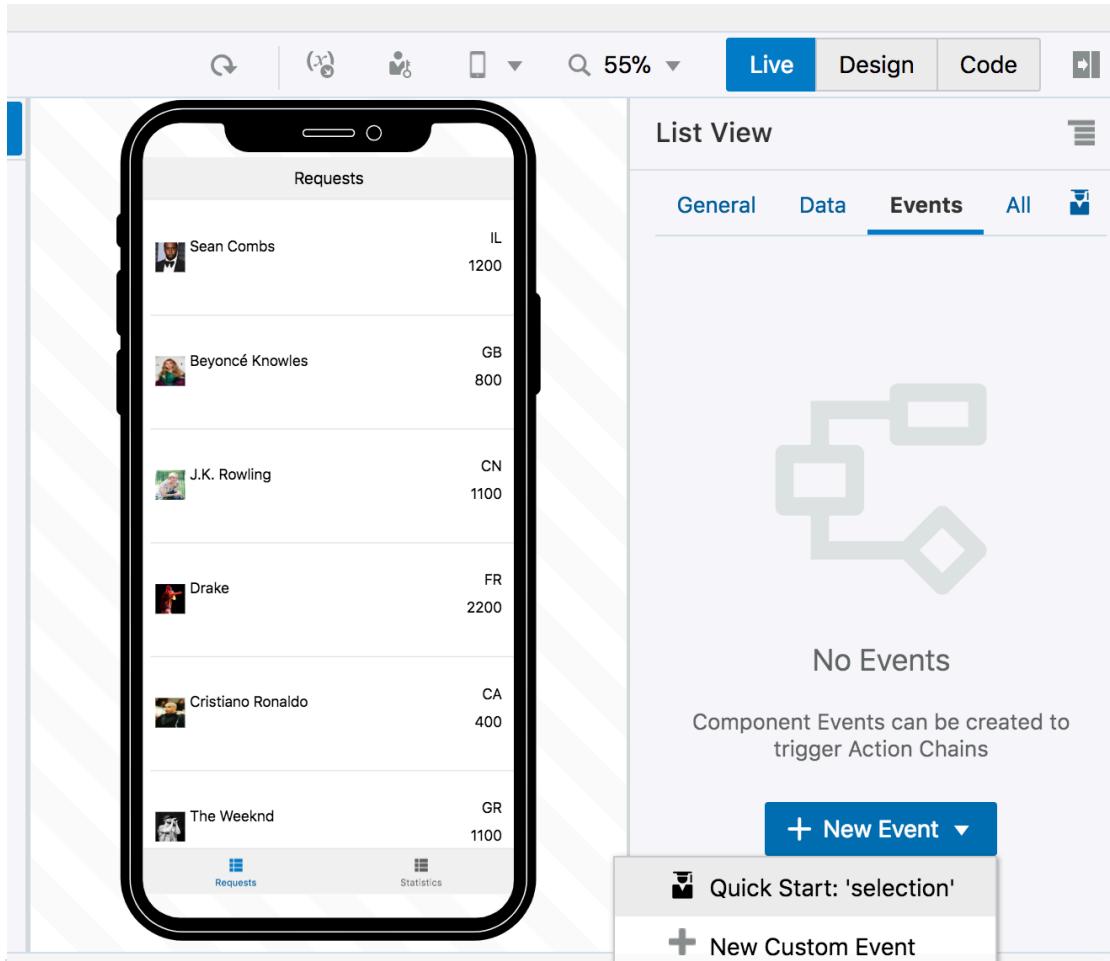
- Mobile Page Template
  - Flex Container (List View)
    - List View
    - contextMenu
    - itemTemplate
      - Row Template
        - List Item
        - Image
        - Image

**List View**

General Data Events All

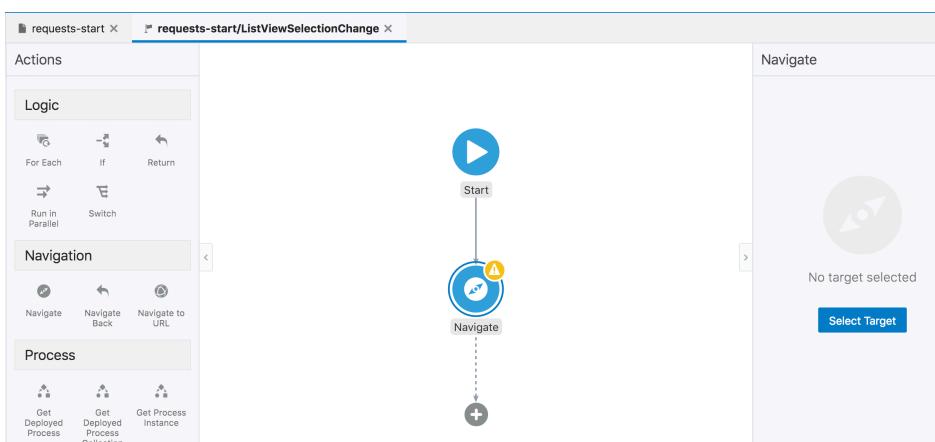
Selection Mode: Single

- Click the **Events** tab in the Property Inspector. Click **New Event** and select **Quick Start: selection**.

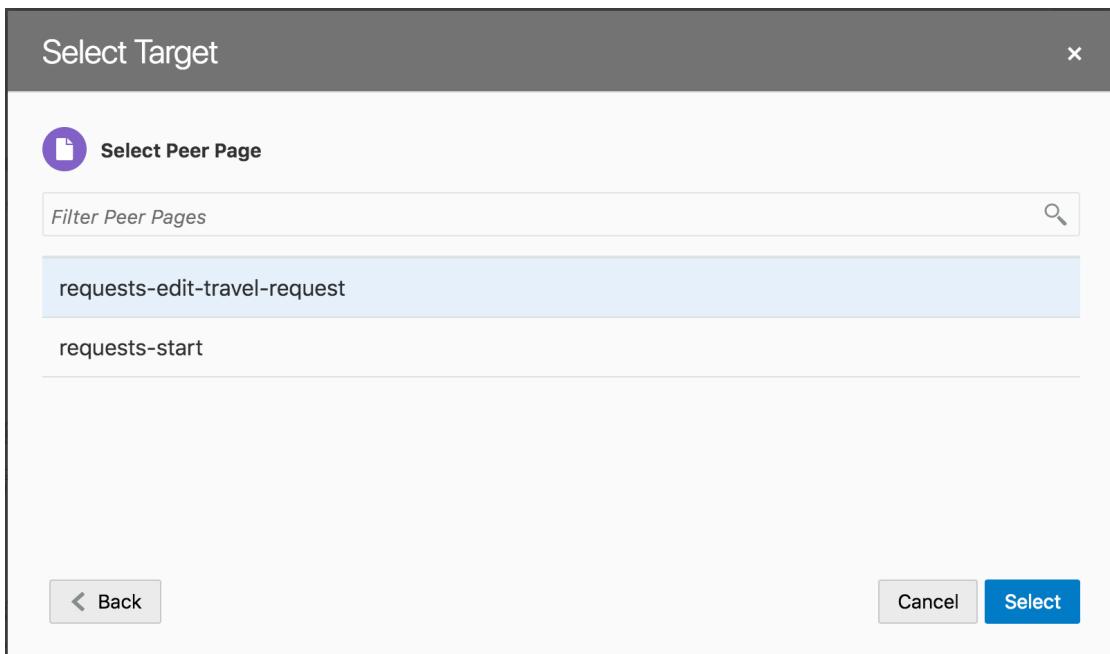


Now you are in the **Action Chain Designer** – where you can map the exact event process that you want to happen when the user selects a row in the list

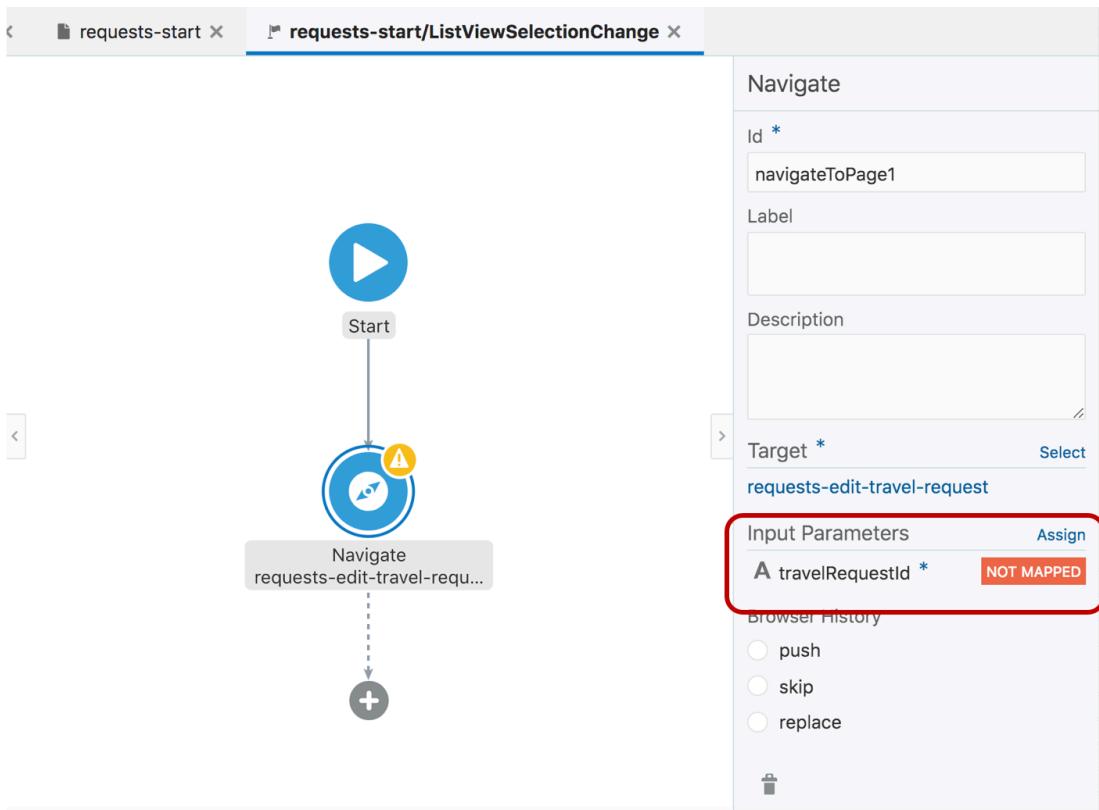
- Drag a **Navigate** component onto the editor. In the Property Inspector, click **Select Target**.



- Click **Peer Pages** > **requests-edit-travel-request** and click **Select**.



6. The Edit page has a mandatory input parameter called **travelRequestId** that tells it which record to load. You need to pass in this parameter when you navigate to the page. Click **travelRequestId** in the **Property Inspector**.



7. Map **Action Chain > selection > item[0]** on the left to **travelRequestId** on the right.

The screenshot shows the 'Navigate' dialog in Oracle Visual Builder. The 'Sources' section on the left lists various types like Action Chain, Page, and Flow. Under Action Chain, 'selection' is expanded, and 'item[0]' is selected. A blue line connects this to the 'Parameters' section on the right, where 'travelRequestId \*' is listed under 'Parameters'. Below this, an expression '\$chain.variables.selection[0]' is defined. At the bottom right are 'Cancel' and 'Save' buttons.

## 8. Click Save.

### Running the Application

You can run the application either in **Live** mode in the **Page Designer** or in a separate preview window.

1. To run in a separate window, click the **Run** button in the top right of the designer.

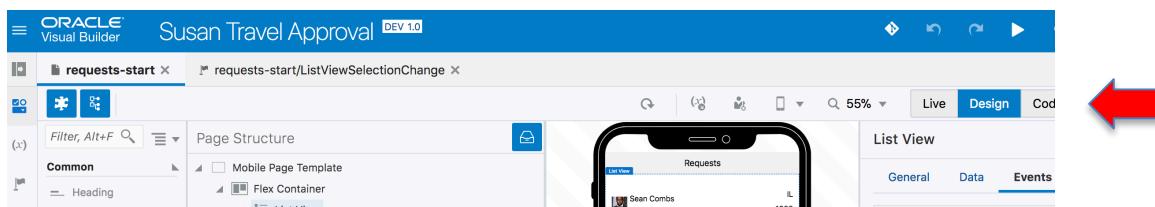


2. You can view how the mobile app will render for different device types by selecting the **mobile device type** in the dropdown at the top of the window.
3. Update a few of the records:

**Sean Combs – El Al, Approved**  
**Beyonce, JK Rowling – British Airways**  
**Drake – KLM, Approved**  
**Christiano Ronaldo – KLM, Approved, Cost 2000**

You can also run the application in the Page Designer by switching to Live mode.

4. In the Designer, open the **requests-start** tab and click the **Live** button. Then select a record. The Edit page opens and loads the record you selected.



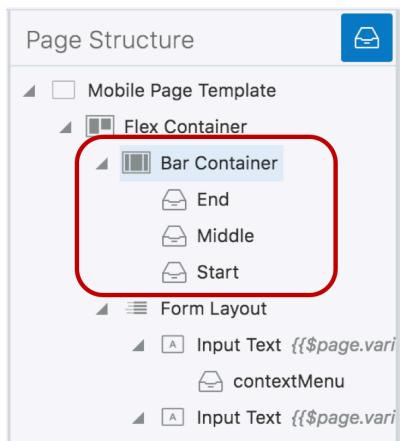
5. Return to Design mode by clicking the **Design** button

## Modifying the Edit Page

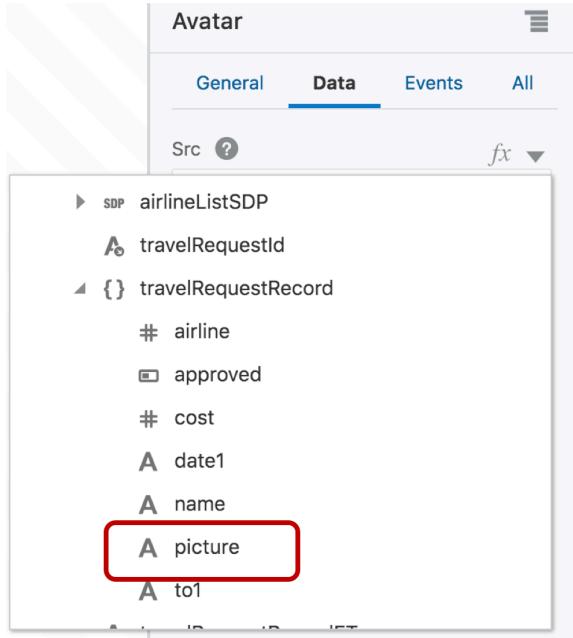
Now you will modify your edit page to make it look a bit better and to display some information about the country that the Travel Request took place in. Here's what your completed page will look like:



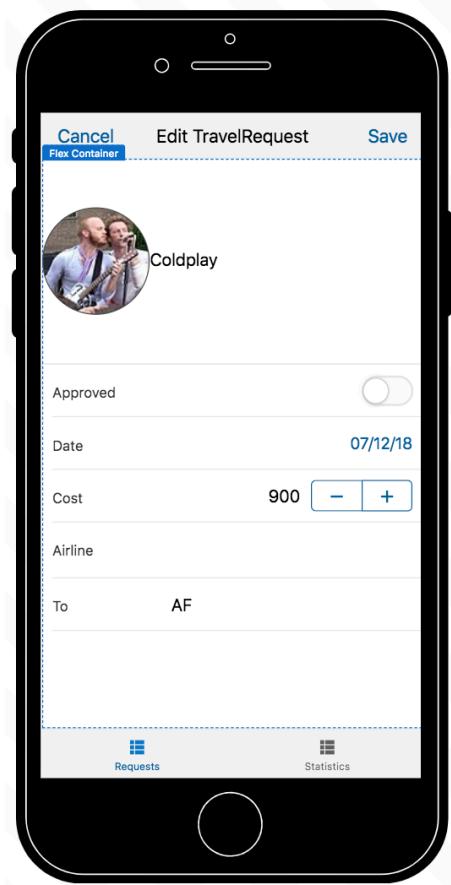
1. Open the Page Structure editor. It's often easier to drag and drop onto the Page Structure than onto the Page Designer.
2. Drag a Bar Container into the Flex Container and place it above the Form Layout component.



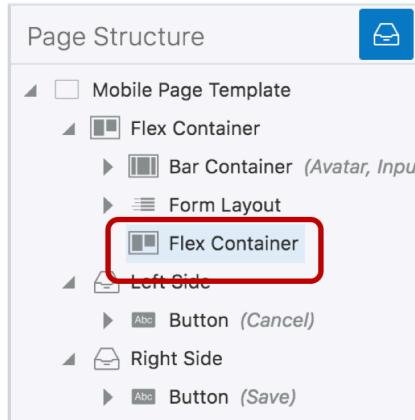
3. Drag an **Avatar** component into the **Start** slot in the **Bar Container**.
4. In the Property Inspector, set the Size to LG. Click the Data tab and set the Src property to travelRequestRecord > picture.
5. Delete the picture (text field) from your page



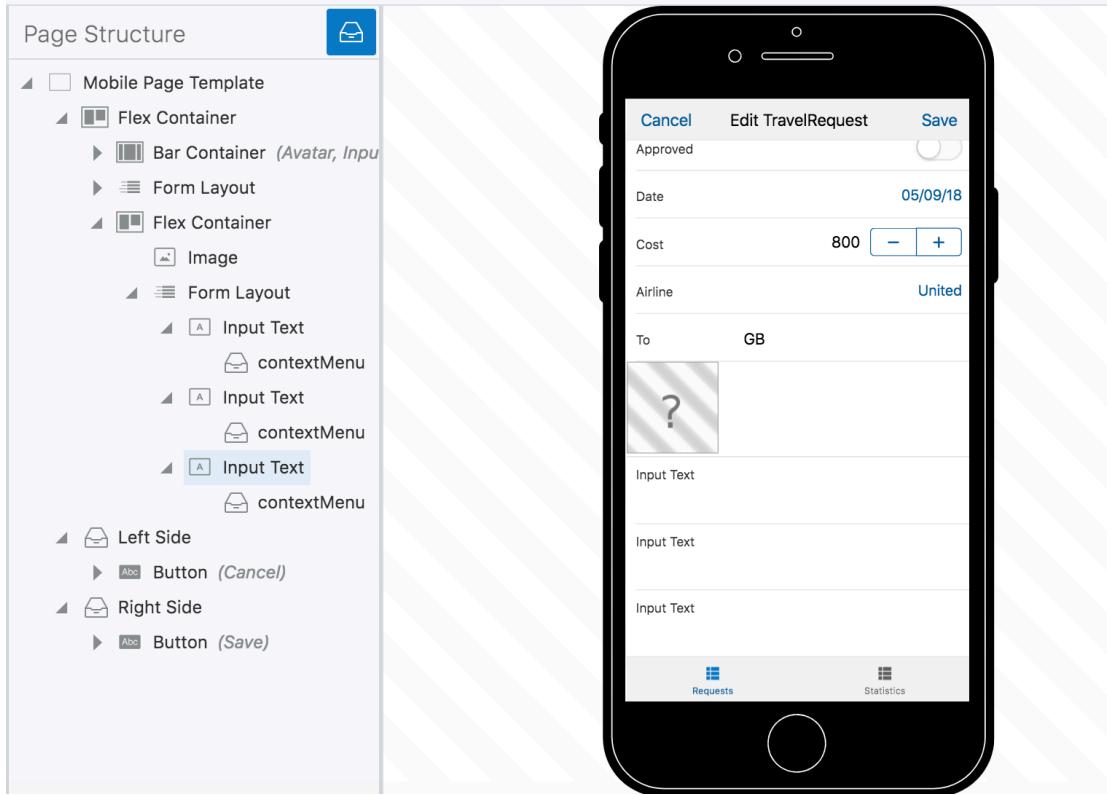
6. Grab the **Name** field (in the page) and drag it up to the **End** slot in the **Bar Component**. Note that you have to grab the value for the field ("Coldpay") and not the label "**Name**". Your app should now look like this:



7. Drag another **Flex Container** below the **Form Layout** but still inside its parent **Flex Container**.

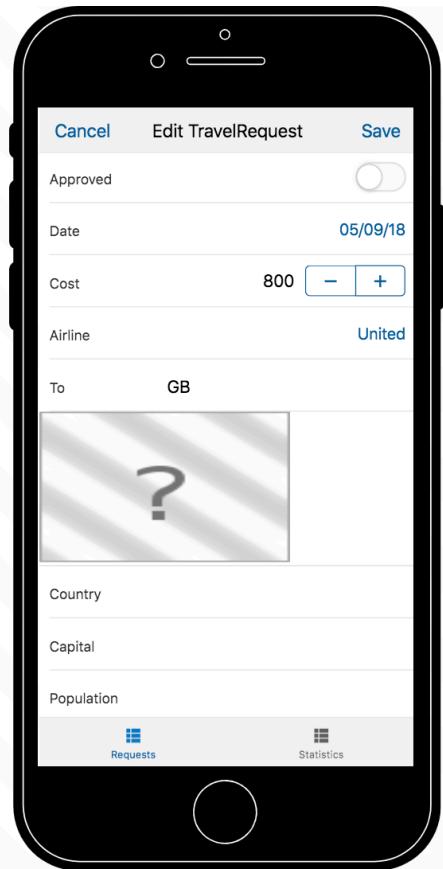


8. Place an **Image** component into the **Flex Container** you just added.
9. Place a **Form Layout** below the Image.
10. Place three **Input Text Components** into the **Form Layout**. Your page should look like this:



11. Select the **Image** component. In the **Property Inspector**, set the **Width** to **250** and the **Height** to **150**.
12. Select the **Form Layout** and set **Label Edge** to **Start**.
13. Change the labels on the Input Fields by selecting each field and changed in the **Label Hint** property in the **Property Inspector**. Change the fields to:
  - **Country**
  - **Capital**
  - **Population**

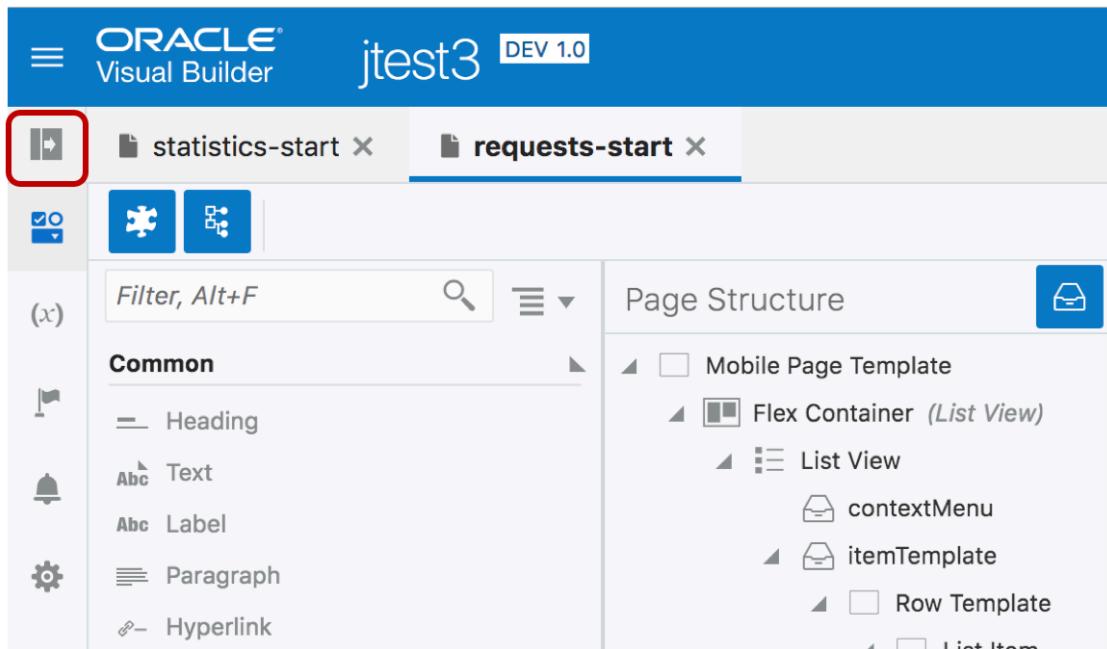
Your app should look like this:



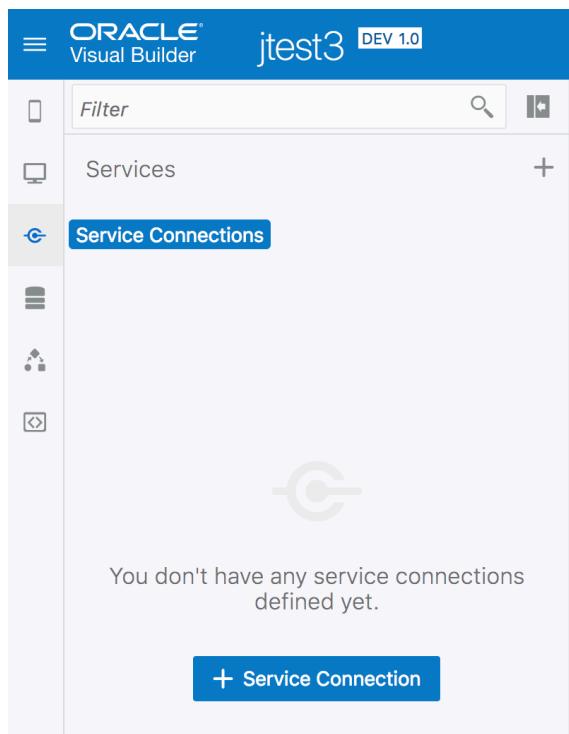
Now you are going to call an external Rest Service from this page to provide some additional information about the country to be visited.

### Registering an External REST Service

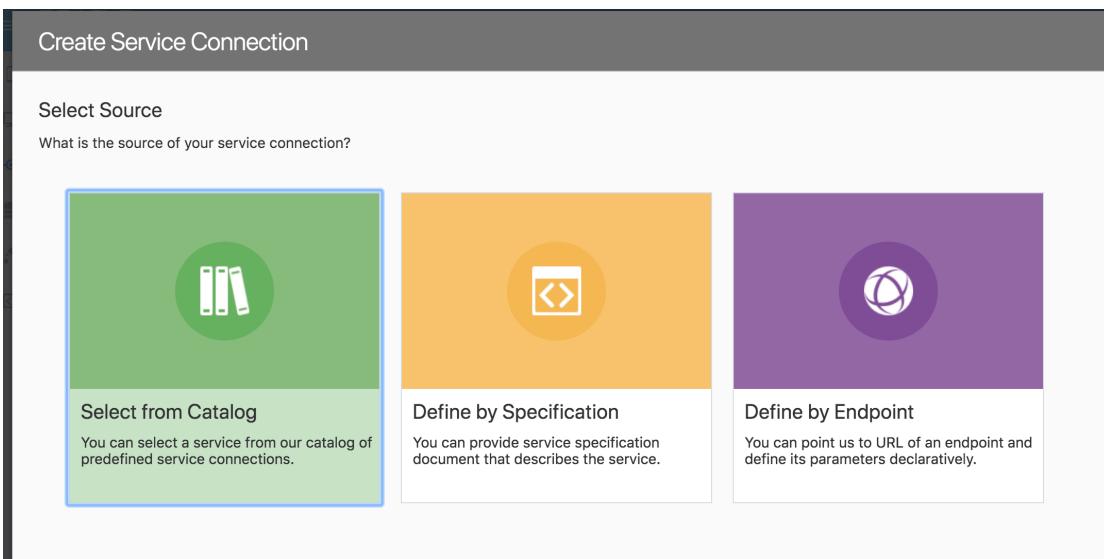
1. If it isn't already open, open the Application Navigator by clicking the  icon on the left.



2. Click **Service Connections** in the right Application Navigator to open the **Services** window.



3. Click **+ Service Connections > Define By Endpoint**.



#### 4. Set URL to <https://restcountries.eu/rest/v2/alpha/{code}>

This is the REST API that returns the info on a country. Since it returns an object rather than an array we need to tell VBCS to treat it as a Get One action.

#### 5. Set Action Hint select **Get One**.

Method \* URL \*

Method: GET URL: https://restcountries.eu/rest/v2/alpha/{code}

Action Hint: Get One

Start by choosing the HTTP method and giving us the URL for your endpoint. It can optionally include query parameters.

#### 6. Click Next.

#### 7. Under the **Service** tab, set **Service Name** to [GetCountry](#).

#### 8. Click the **Test** tab. In the **URL Parameters** section, add a value of **US** to the **code** parameter and click **Send**.

The screenshot shows the VBCS interface with the 'Test' tab selected. At the top, the method is set to 'GET' and the path is '/alpha/{code}'. Below this, there are tabs for Service, Authentication, Request, Response, and Test. Under the Request tab, there are sections for Body and Headers, and a 'URL Parameters' section where a parameter named 'path' is mapped to 'code' with a type of 'string' and a value of 'US'. A 'Required' checkbox is checked. There is also a 'Reset to Defaults' button. A 'Send' button is available at the bottom left, and the status is shown as 'Status:'. The Response section is collapsed.

VBCS calls the API and the response is shown in the Response section.

## 9. Click **Copy to Response Body**.

10. Click the **Response** tab. This shows the format of the response that you copied from the **Test** tab that will be returned by the **GetCountry** service. VBCS stores the response body so it knows how to deal with responses to this REST call.

The screenshot shows the 'Create Service Connection' dialog with the 'Test' tab selected. The configuration is identical to the previous screenshot: Method 'GET', Path '/alpha/{code}', Action Hint 'Get One', and URL Parameters 'path' mapped to 'code'. The Response tab is selected, showing the response body as a JSON object: { "name": "Bulgaria", "topLevelDomain": [ "bg" ] }. A 'Copy to Response Body' button is visible next to the JSON. The status is 'Status: 200 OK'.

11. Click **Create** to create the service.

## Creating a Variable Based on the Service

In order to use the data returned in the REST Service call, you create a Type based on the format of the data returned. Then you create a variable based on that Type to use in your pages.

1. Click the **Variables** icon of the [requests-edit-travel-request](#) page, and select the **Types** tab.

The screenshot shows the SAP Fiori Launchpad with the 'requests-edit-travel-request' application open. The 'Types' tab is selected. Three service types are listed:

- {} businessObjectServiceErrorResponseType
- {} getallAirlineResponse
- {} updateTravelRequestRequest

For each service type, there are two buttons: 'Add Field' and 'Edit from Endpoint'.

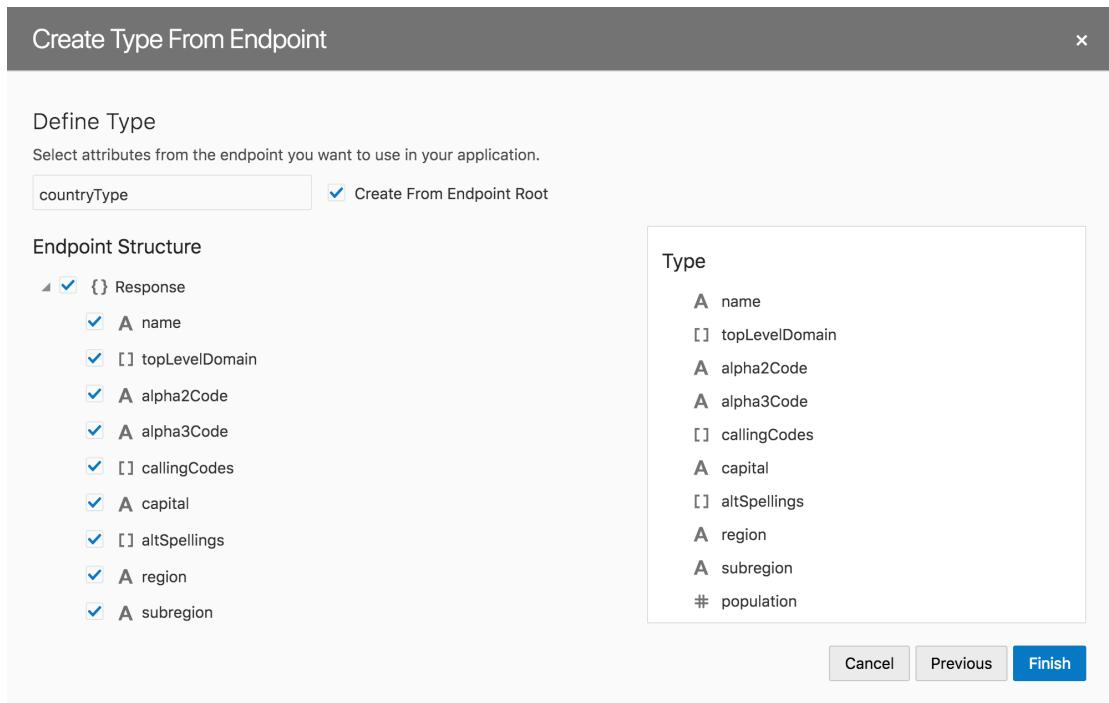
2. Click on the **+ Type** dropdown and select **From Endpoint**

3. Expand Service Connections, **GetCountry** and select **GET**

The screenshot shows the 'Create Type From Endpoint' dialog box. Under 'Service Connections', the 'GetCountry' endpoint is expanded, showing a 'GET /alpha/{code}' option which is currently selected. At the bottom right of the dialog are 'Cancel' and 'Next' buttons.

4. Click **Next**.

5. Call the type **countryType** and check the **Response** checkbox to select all the attributes returned by the service

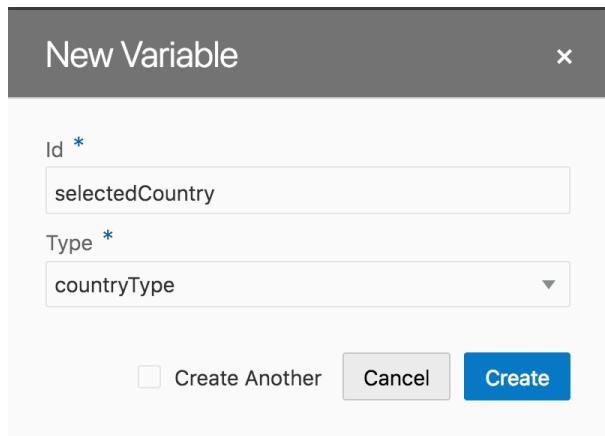


6. Click **Finish**.

7. Click the **Variables Tab** to create a **Variable** based on **countryType**.

8. Click **+ Variable**.

9. Set the **Id** to **selectedCountry** and the **Type** to **countryType**.



10. Click **Create**.

### Connect the Page Fields to the Variable

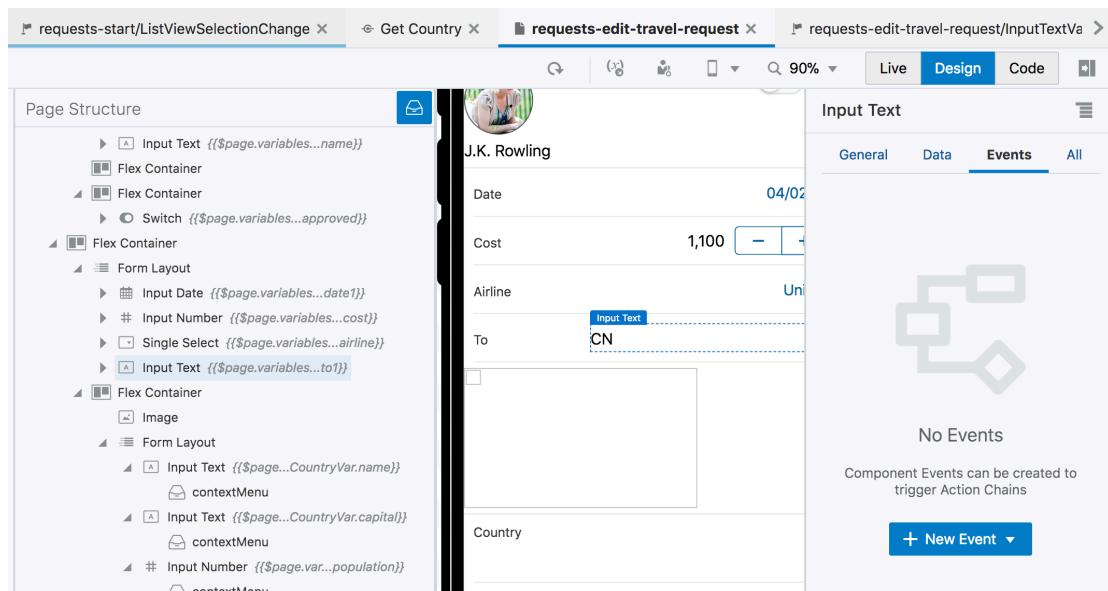
1. Return to the Page Designer.

2. Select the **Image**. In the **Data** tab of the **Property Inspector**, set the Source URL to **selectedCountry > flag**.

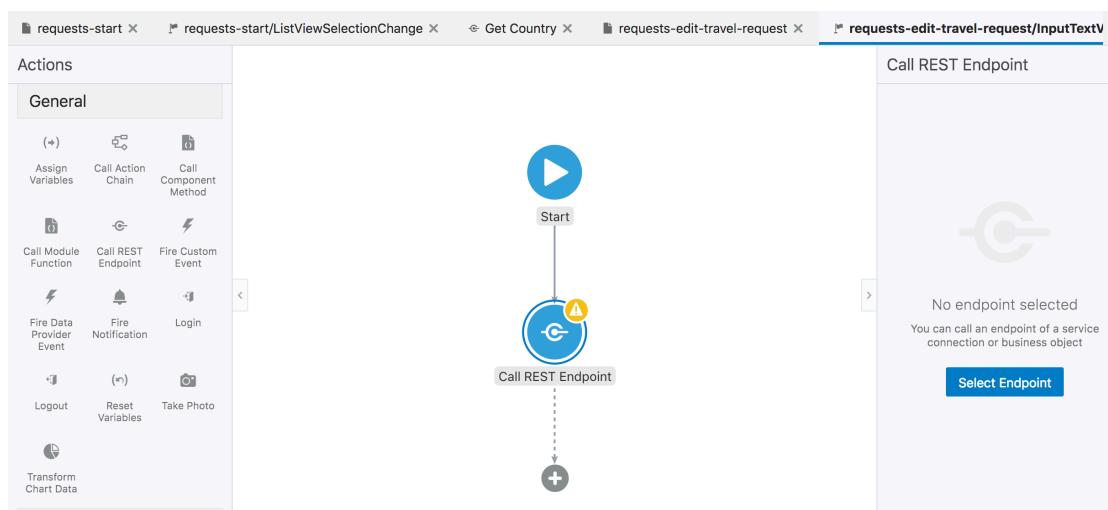
3. Select the Country field in the Form Layout below the image. In the **Data** tab of the **Property Inspector**, set the **Value** to **selectedCountry > name**.
4. Do the same to bind **Capital** to **selectedCountry > capital** and Population to **selectedCountry > population**.

## Calling the REST API

1. Select the **To** field in the Page Designer. In the Property Inspector, click the **Events** tab.

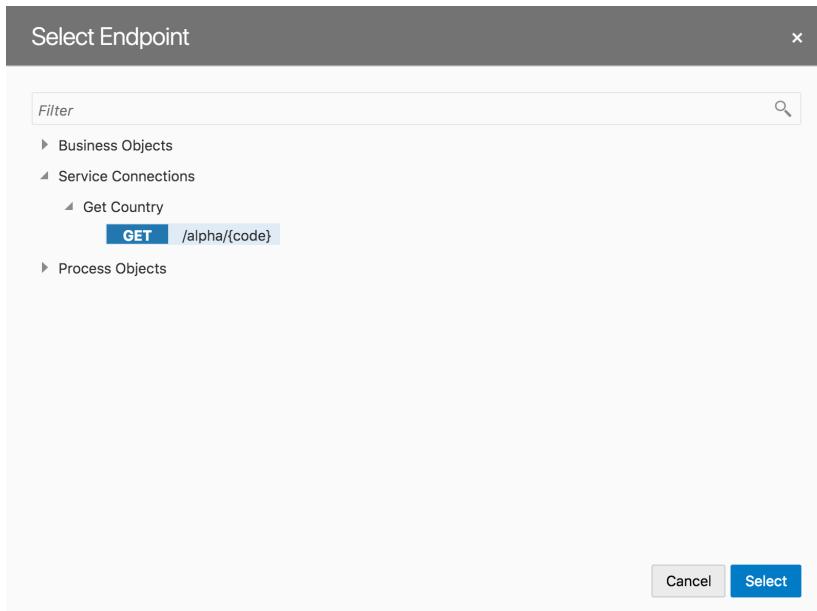


2. Click **New Event > Quick Start: 'value'**. VBCS creates an event for you on the value change of the **To** field. It also creates and opens the **InputTextValueChange** action chain.
3. From the **Actions** palette, drag **Call REST Endpoint** onto the action chain.

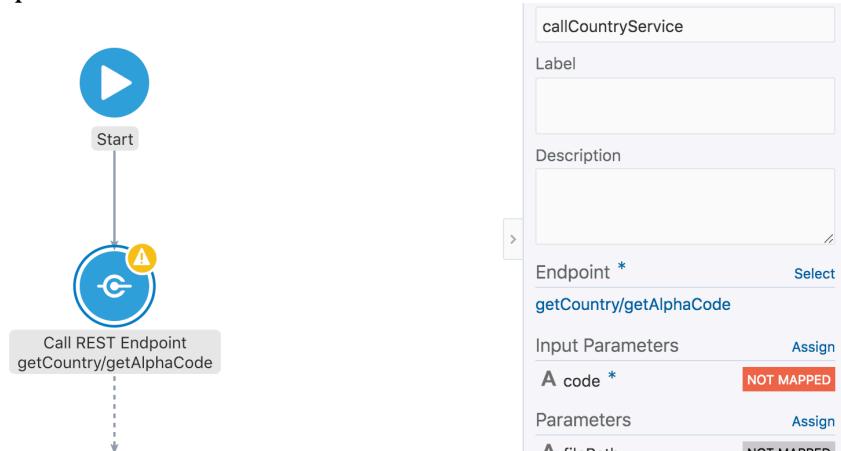


4. Click **Select Endpoint**.

5. Select **Service Connections** > **GetCountry** > **Get /alpha/{code}** and click **Select**.



6. In the Property Inspector, set the **ID** to **callCountryService**.
7. Now you need to pass in **travelRequestRecord.to1** as the **code** input parameter. Under Input Parameter click **code**.



8. Map **Page > travelRequestRecord > to1** on the left to **Parameters > code** on the right.

Map Variables To Parameters

```
▶ {} CountryInfo
  A travelRequestId
  ▲ {} travelRequestRecord
    # airline
    ☐ approved
    # cost
    A date1
    A from1
    A to1 ——————>
    A traveler
    A travelRequestRecordETag
```

Expression for: code

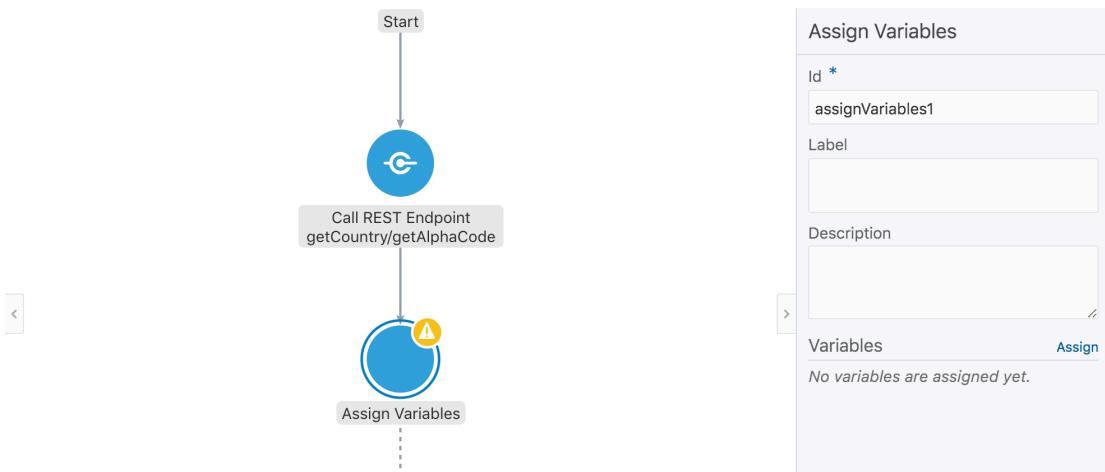
```
{{ $page.variables.travelRequestRecord.to1 }}
```

Target Filter

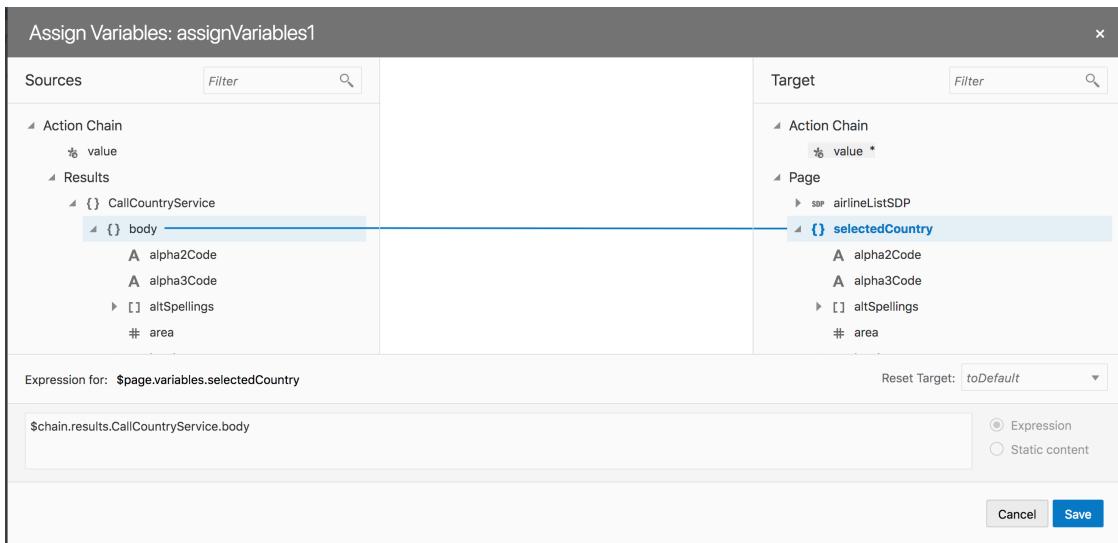
Parameters

A code \*

9. Click **Save**.
10. Now you map the data returned by the call to the REST Service to the variable based on the response data format. Drag and drop an **Assign Variable** action below the **Call REST Service** action.



11. In the **Property Inspector**, click **Assign**.
12. Map **Results > CallCountryService > body** on the left to **Page > selectedCountry** on the right.



### 13. Click **Save**.

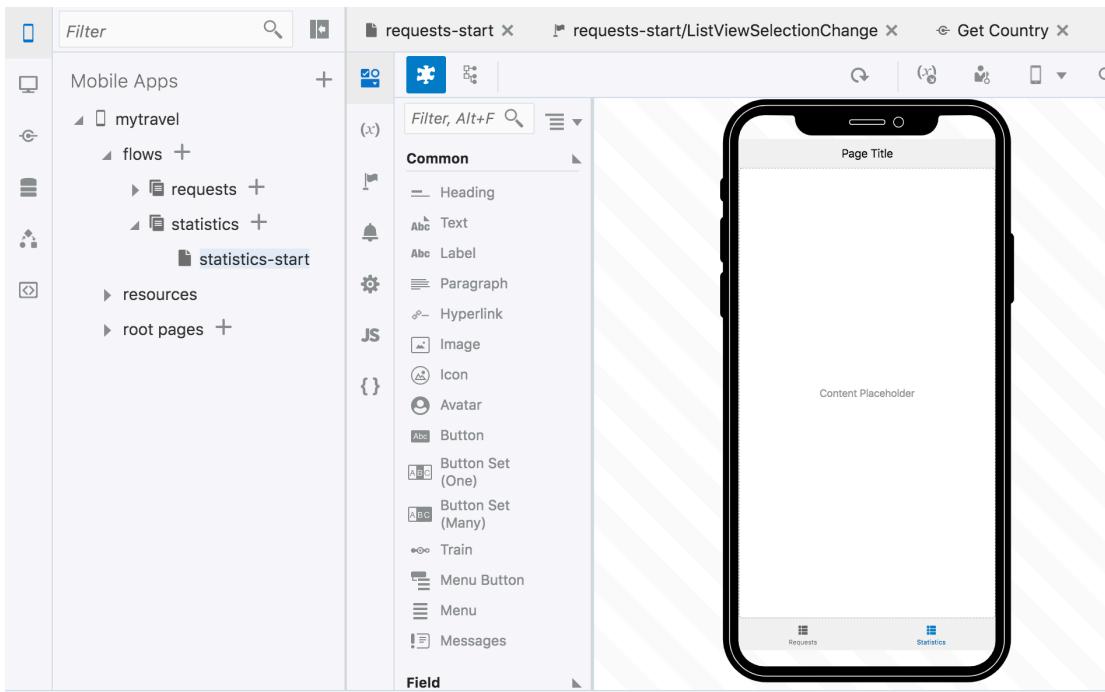
## Running the Completed Application

1. Click **Run** to run the application.
2. Select a row and click to open the **Edit TravelRequest** page.
3. Change the destination to another country (ISO code, i.e. IT or US) and ensure that the country field and data update.

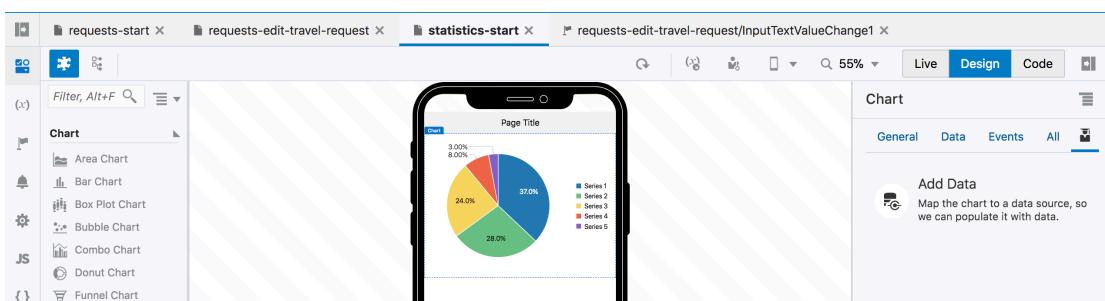
## Creating Visualizations of Data

In this section you will add some visual components to the **Statistics** tab that you created at the start of this lab.

1. Expand the main **Application Navigator**, open the Mobile Apps tab, and click **myTravel** > flows > statistics > statistics-start.

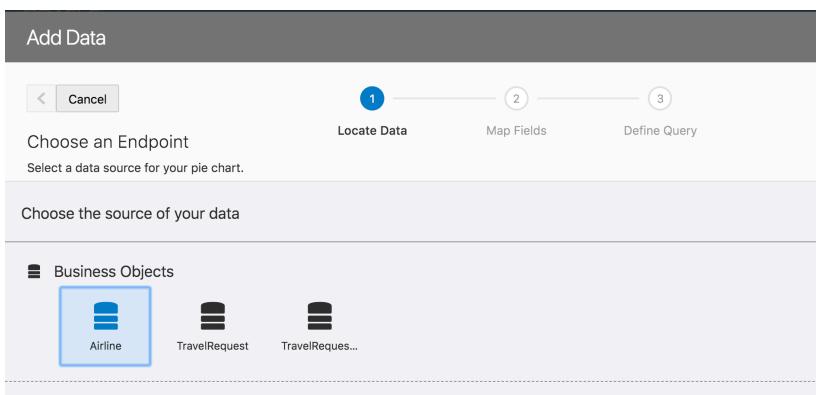


2. Drag a **Pie Chart** from the **Component Navigator** onto the page.



3. Click **Add Data**.

4. Choose the **Airline** business object and click **Next**.



5. Drag **totalCost** into the Slice Values.

6. Drag **airline** into the Slice Colors.

7. Click **Next**, then click **Finish**. Data from the **Airline** business object is immediately rendered on the chart. Note that if you did not edit some records to specify an Airline then there is no data to display here.
8. In the Property Inspector for the chart, click the **General** tab and change the **animation-on-display** to **zoom**.
9. Drag a **Scatter Chart** onto the page below the pie chart.
10. Click **Add Data**.

11. Choose the **Airline** business object and click **Next**.
12. Drag **totalCost** onto the **x Axis**.
13. Drag **averageCost** onto the **y Axis**.
14. Drag **airline** onto **Marker Colors**.

Add Data

Map Fields

Drag and drop fields to display for your scatter chart.

Endpoint Structure

Filter

- items
  - A airline
  - # averageCost
  - A createdBy
  - creationDate
  - # id
  - lastUpdateDate
  - A lastUpdatedBy
  - links
  - # totalCost

Scatter Chart Component Fields

\* Values (X Axis)

\* Values (Y Axis)

\* Marker Colors

Marker Markers

Primary Key

Next >

15. Click **Next**, then click **Finish**.
  16. Click the header at the top of the page to select the page template. In the Property Inspector, set **Page Title** to **Airline Statistics**.
  17. Drag a **Heading** above each of the charts and name them:
- Pie Chart – **Total Spend**
  - Scatter Chart – **Cost Analysis**

Your app should look something like this:

Page Structure

Mobile Page Template

- Flex Container
  - Heading (*Total Spend*)
  - Chart
    - contextMenu
    - groupTemplate
    - itemTemplate
      - Template
    - seriesTemplate
  - Heading (*Cost Analysis*)
  - Chart
    - contextMenu
    - groupTemplate
    - itemTemplate
      - Template
    - seriesTemplate
- Left Side
- Right Side

18. Click on the Code tab. You can edit all the elements in your page from here if you prefer – for instance, find the **legend.title="Airline"** for one of the charts and edit it to **legend.title="Airline Spend"**



```

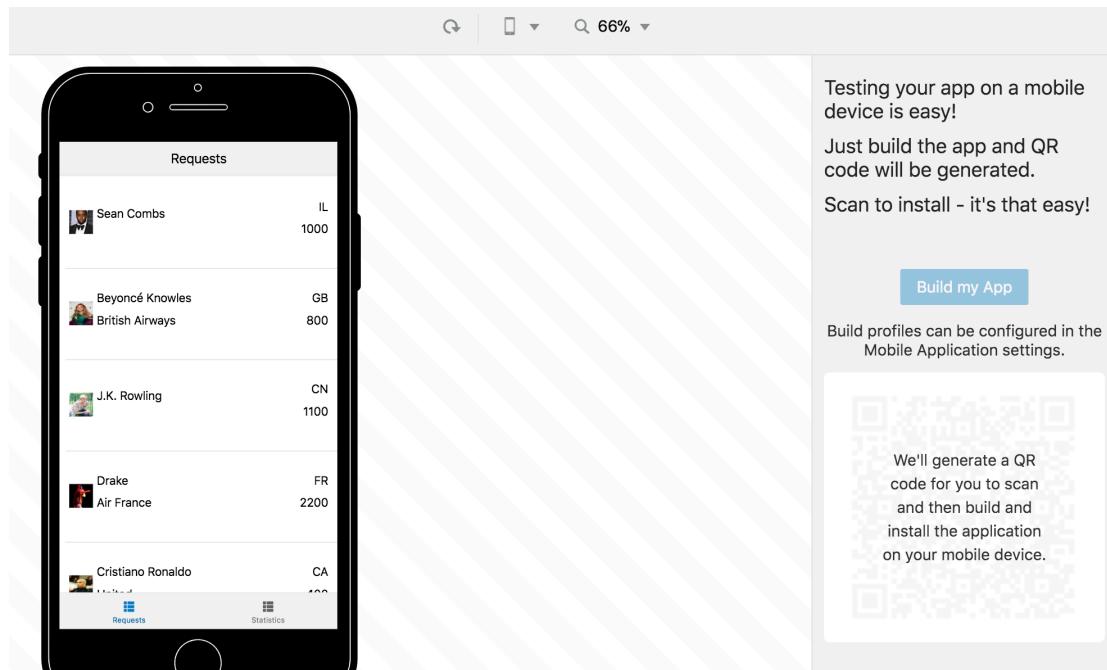
1 <oj-ext-mobile-page-template id="page-template" page-title="Airline Statistics">
2   <div class="oj-flex-item oj-flex oj-sm-flex-direction-column oj-sm-flex-wrap-nowrap">
3     <h2 id="h1-729287828-1" class="oj-flex-item">Total Spend</h2>
4     <oj-chart type="pie" class="oj-flex-item oj-sm-flex-0" style="max-width:100%;" id="oj-chart-729287828-1"
5       data="[$page.variables.airlineListSDP]" track-resize="off" legend.title="Airline"
6       animation-on-data-change="auto" animation-on-display="zoom">
7       <template slot="itemTemplate">
8         <oj-chart-item value="[$current.data.totalCost]" series-id="[$current.data.airline]" group-id="[['Undefined']]"></oj-chart-item>
9       </template>
10      </oj-chart>
11    </div>
12    <div class="oj-flex">
13      <h1 id="h1-729287828-2" class="oj-flex-item">Cost Analysis</h1>
14      <oj-chart type="scatter" class="oj-sm-flex-initial oj-flex-item" style="max-width:100%;">
15        id="oj-chart-729287828-2" data="[$page.variables.airlineListSDP]" track-resize="off"
16        x-axis.title="totalCost" y-axis.title="averageCost" legend.title="airline">
17        <template slot="itemTemplate">
18          <oj-chart-item x="[$current.data.totalCost]" y="[$current.data.averageCost]" series-id="[$current.data.airline]" z="1" group-id="[['Undefined']]"></oj-chart-item>
19        </template>
20      </oj-chart>
21    </div>
22  </oj-ext-mobile-page-template>

```

This concludes the main part of this Lab. Congratulations, you have created your first mobile application complete with Business Objects, a REST Service call and visualizations – all done in the **Visual Builder Cloud Service** primarily declaratively, but with access to the code if you prefer to work at the code level.

### If You Have Extra Time

In this lab you have run your app using the built-in mobile application simulator. You will have noticed that the 'Build My App' option was disabled



Before you can build the app for installation on an iOS or Android device, you have to create a Build Configuration

Below follows an example of how to setup and build an iOS version of your mobile app. The lab does not provide the elements for you to complete this setup. This section is for information only

1. Click on the **Settings** icon for your mobile app and click on the **Build Configurations** tab

The screenshot shows the VBCS interface with the 'Build Configurations' tab selected. On the left, there's a sidebar with icons for General Settings, Build Configurations (selected), Security, and Translations. Below the tabs, there's a table header with columns: OS, Type, Name, and Used for. A message says 'No data to display.' A red arrow points to the 'New Configuration' button at the top right.

2. Click **New Configuration**

3. To follow this example, click on **iOS**, but if you are more familiar with Android, go there too

The screenshot shows the VBCS interface with the 'Build Configurations' tab selected for the 'mytravel' app. On the left, there's a sidebar with icons for Mobile Apps, flows, resources, and root pages. Below the tabs, there's a table header with columns: OS, Type, Name, and Used for. A dropdown menu labeled 'New Configuration' shows 'Android' and 'iOS', with 'iOS' highlighted. A red arrow points to the 'iOS' option in the dropdown.

4. The configuration settings on the left side are defaulted by VBCS, but can all be edited to suit your needs

5. On the right side you must add your Apple Developer program credentials – whether an Enterprise or Standard program, developer or distribution profile etc.

Note: it is outside the scope of this lab to provide instruction on how to get an Apple Developer Program account. Go to [developer.apple.com](http://developer.apple.com) for more details

6. Add your provisioning profile, certificate and certificate password details to the configuration. Ensure that the Signing Identity is exactly as shown in your MAC Keychain entry for the certificate. To ensure this, use Get Info in keychain and copy the complete entry from the Common Name property

7. If you want to use this profile as the default build when you ask VBCS to build your mobile app, click on the appropriate checkboxes

**iOS Build Configuration**

---

Configuration Name * <input type="text" value="susanTravel"/> Build Type <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">Release</div> Bundle ID * <input type="text" value="vbcs.mytravel.Susan Travel Approval.bundleid"/> Bundle Version Name * <input type="text" value="1.0.0"/> Bundle Version * <input type="text" value="1.0.0"/>	Certificate * <div style="border: 1px dashed #ccc; padding: 5px; position: relative;"> <div style="text-align: right; margin-bottom: 5px;"><a href="#">Test.p12</a> </div> <input type="password" value="*****"/> </div> Certificate Password <input type="password" value="*****"/> Provisioning Profile * <div style="border: 1px dashed #ccc; padding: 5px; position: relative;"> <div style="text-align: right; margin-bottom: 5px;"><a href="#">Test.mobileprovision</a> </div> <input type="text" value="iPhone Distribution: Test"/> </div> Signing Identity * <input type="text" value="iPhone Distribution: Test"/> <div style="margin-top: 10px;"> <input checked="" type="checkbox"/> Default Configuration for Publish  <input checked="" type="checkbox"/> Default Configuration for Stage         </div>
--	--

---

8. Now when you run the app the Build My App button will be enabled.

9. Clicking on it takes you to Stage Application. Here you can select how the business data should be deployed:

If this is a first Stage deployment you may want to take the development data into the app, or start with a clean database. For subsequent deployments you may want to keep any data that was in the previously staged version, start from clean or start from the development data again.

**Stage Application**

---

Your application in Development will be deployed to Stage.

Business Object Data:

- Keep existing data in Stage
- Stage application with a clean database
- Replace Stage data with Development data

All Business Object data will be copied from Development to Stage.

---

11. Once you hit the Stage button the Mobile application is staged and you can install on your device (via the QR scan or by downloading the .ipa file) and start the next stage of your development process

## 12. You can have multiple build configurations for iOS and Android releases

Testing your app on a mobile device is easy!

Just build the app and QR code will be generated.

Scan to install - it's that easy!

[Rebuild my App](#)

[iOS](#)   [Android](#)

iOS

[Download for iOS](#)

Name	Country	Value
Cristiano Ronaldo	CA	400
Drake	FR	2200
Beyoncé Knowles	GB	800
LeBron James	GR	1800
James Patterson	US	800
Coldplay	AF	900