## **Evaluate and Implement Dynamic Dashboard Layout Configuration with Forge Support for Charts**

### 

### **Objective**

Enable customizable dashboards in ODC Studio with support for drag-and-drop tile arrangement, user-specific layout persistence, and chart integration. The layout must be dynamic, persistable, and reloadable.

## 

## Spike Summary: Goals and Validation

### 

### **Problem Addressed**

Current dashboards are fixed. This guide solves:

* Reordering tiles with drag-and-drop
* Hiding/showing user-specific content
* Per-user layout persistence
* Integration of charts per tile with configuration

### 

### **Key Questions Answered**

| Question | Answer |
| --- | --- |
| Can we persist layout per user? | Yes, using DashboardLayout entity + UserId |
| Are there Forge plugins for layouts and charts? | Drag and Drop, Layout Grid, OutSystems Charts, HighCharts Plugin |
| Can layout versioning be supported? | JSON includes version field per user |

### 

### Developer Tasks Covered

| Task | Implemented in Guide |
| --- | --- |
| Build settings UI with drag-and-drop | Step 5 |
| Hide/show tiles | Step 5 (visibility control) |
| Add tile types and persist layout | Step 4 and Step 6 |
| JSON layout model per user | Step 4 |
| REST-like API for load/save | Step 6 via Server Actions |
| Integration of chart plugin | Step 2 and Step 5 |
| Reload and apply saved layout | Step 7 |

### 

### **Deliverables Checklist**

| Deliverable | Met? |
| --- | --- |
| Drag-and-drop dashboard UI | Ok |
| JSON metadata and persistence | Ok |
| REST-like persistence interface | Ok |
| Integrated charting component | Ok |
| Plugin evaluations | Ok (Forge components tested) |
| Working demo (Preview) | Ok |
| Layout versioning notes | Ok version field per layout |

**Step-by-Step Implementation Guide**

### **Step 1: Create a Reactive Web App in ODC Studio**

1. Open ODC Studio from your browser (e.g., studio.outsystems.com).
2. Click the “New Application” button on the dashboard.
3. In the modal that appears, select Reactive Web App as the application type.
4. Enter the application name (e.g., DynamicDashboard) and optional description.
5. Click Create Application.
6. Once generated, you’ll be taken to the default module with a screen named Home.
7. Click Preview App (top right corner) to verify the application opens correctly.
8. Close the preview and return to design mode.

### **Step 2: Install Required Forge Components**

1. On the left sidebar, click Forge.
2. Search and install the following components one by one:
   * Drag and Drop (for tile rearrangement)
   * Layout Grid (for structured layout)
   * Charts Library (e.g., OutSystems Charts or HighCharts Wrapper)
3. After clicking Install, wait for the dependency to load and appear under Dependencies.
4. Return to your app flow and click 1-Click Publish to publish changes.
5. Click Preview again to verify everything is loading without errors.

### **Step 3: Create Entity to Store Dashboard Layout**

1. Go to the Data tab.
2. Under Entities, click + Entity and name it DashboardLayout.
3. Add attributes:
   * Id (Auto Number, primary key)
   * UserId (Text): associate with current user
   * LayoutJson (Text): to hold serialized tile data
   * Version (Integer): supports version control
   * UpdatedAt (DateTime): timestamp of last save
4. Click 1-Click Publish to create this data model in the database.
5. Optionally, go to Data > Test Data and create one mock layout row manually for test purposes.

### **Step 4: Define JSON Layout Model**

Structure of the saved layout. It will drive the UI. Example:

{  
 "tiles": [  
 {  
 "tileId": "tile1",  
 "title": "Policy Summary",  
 "row": 0,  
 "column": 0,  
 "width": 6,  
 "height": 4,  
 "visible": true,  
 "chartType": "bar",  
 "chartConfig": {  
 "labels": ["Jan", "Feb", "Mar"],  
 "data": [150, 200, 180]  
 }  
 },  
 {  
 "tileId": "tile2",  
 "title": "Premium Breakdown",  
 "row": 0,  
 "column": 6,  
 "width": 6,  
 "height": 4,  
 "visible": true,  
 "chartType": "pie",  
 "chartConfig": {  
 "labels": ["Auto", "Home", "Life"],  
 "data": [50, 30, 20]  
 }  
 }  
 ]  
}

You can extend this with other properties like backgroundColor, refreshInterval, tileType.

### **Step 5: Build DashboardSettings Screen (Admin Layout Editor)**

1. In Interface > UI Flows, add a new screen and name it DashboardSettings.
2. Drag a Layout Grid component onto the canvas. Configure with 12 columns.
3. Add a local variable LocalTiles (data type: Text or Record List based on JSON structure).
4. Use a JSON Deserialize logic to transform LayoutJson from DB into LocalTiles list.
5. Inside the Layout Grid:
   * Add a For Each loop to display each tile.
   * Within the loop, place a Container and bind its position (style) using tile.row and tile.column.
   * Add a Title (Text widget).
   * Add a Chart component and bind chart type and chartConfig.
   * Add a Toggle or Checkbox to hide/show tile (bind to tile.visible).
6. Wrap each tile in the DragAndDrop Forge wrapper. Configure drag zones and drop targets.
7. Capture the Drop Event and adjust the tile.row/tile.column values.
8. Below the grid, add a Save Button. On click:
   * Serialize LocalTiles into JSON using JSON Serialize.
   * Call SaveDashboardLayout server action with current user ID and new layout.
   * Optionally, show feedback (e.g., “Layout saved!”).

### **Step 6: Implement Persistence Logic**

1. Create a Server Action named SaveDashboardLayout in Logic > Server Actions.
2. Add input parameters:
   * UserId (Text)
   * LayoutJson (Text)
3. Inside logic:
   * Use Aggregate to check if a row exists for UserId.
   * If it exists, update it using an Update node.
   * If it doesn’t exist, insert a new layout.
4. Create GetDashboardLayoutByUserId action:
   * Input: UserId
   * Output: LayoutJson (Text)
   * Use this to load layout on app load.

### **Step 7: Create MyDashboard Screen (User View)**

1. Create a screen called MyDashboard for final visualization.
2. On OnInitialize event:
   * Call GetDashboardLayoutByUserId(GetUserId()).
   * Deserialize JSON into local variable (LocalTiles).
3. Add a Layout Grid.
4. Inside a For Each loop:
   * Render each tile using its row/column.
   * Show chart (bar, pie, line) as per chartType.
   * Use tile title and chart data from the layout.
   * Skip tiles where visible is false.
5. Preview your app to test changes live.
6. Move some tiles in DashboardSettings, click save, then refresh MyDashboard to confirm it reflects your saved layout.