# Unit 3 – Developing with SAP Extension Suite

Certification: C_CPE_13

Unit 3 – User Interface and Business Logic

# Agenda

- What have we learned so far
- Generating User Interface
- SAP Fiori Elements
- Custom Business Logic
- Error Handling
- Summary points

# Steps involved

1. Initialize full-stack project – Completed (Unit 1, 2)
2. Create the tables – Data Modeling – Completed (Unit 1, 2)
3. Generic handlers – Out-of-the-box CRUD functionality – Completed (Unit 1, 2)
4. Basic UI
5. List Report layout
6. Custom event handling – Business logic
7. Support for external API
8. Connecting to Sandbox

# Steps involved

1. Initialize full-stack project – Completed (Unit 1, 2)
2. Create the tables – Data Modeling – Completed (Unit 1, 2)
3. Generic handlers – Out-of-the-box CRUD functionality – Completed (Unit 1, 2)
4. Basic UI
5. List Report layout
6. Custom event handling – Business logic
7. Support for external API
8. Connecting to Sandbox

# Step 5 – List Report Layout

```
git checkout 4_list_report (Use tab for branch name)
```

New Files

- common.cds

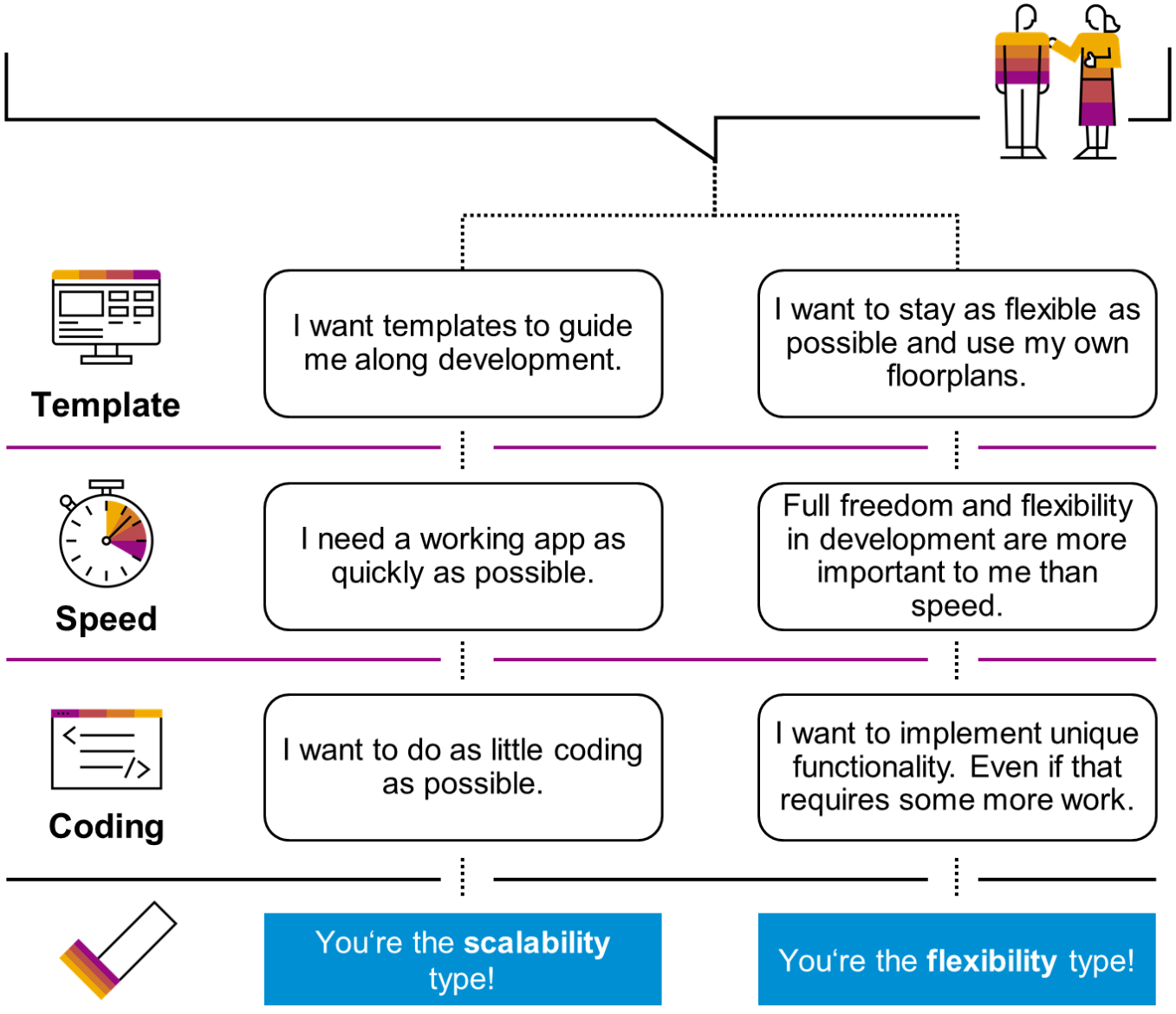
Modified Files

- services.cds
- annotations.cds

Run application command

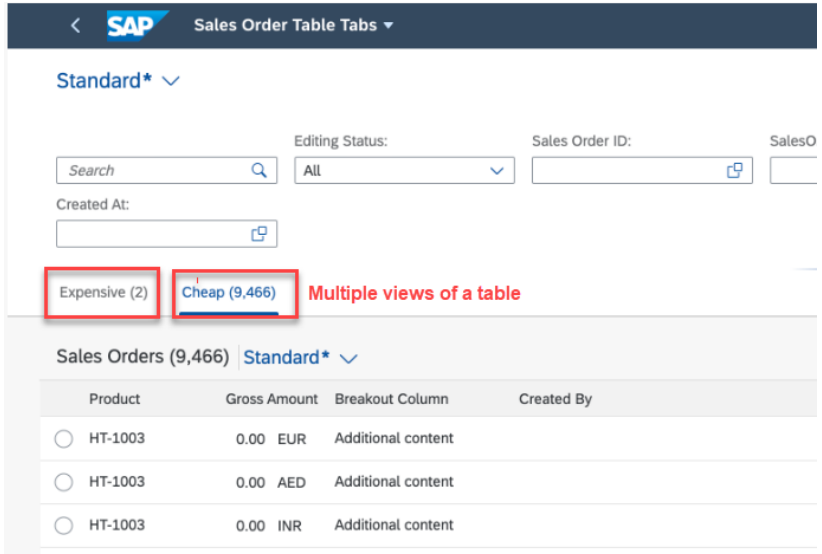
- cds watch

# SAP Fiori Elements Vs SAP UI5 Free Style



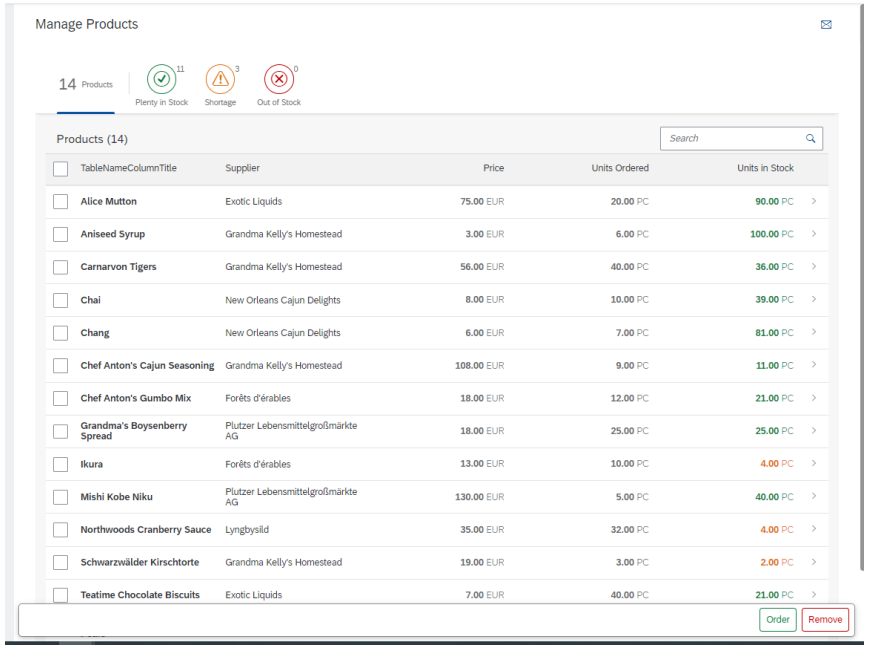
# List Report Floorplan

- Users can view and work with a large set of items
- Searching, filtering, sorting, grouping options are commonly available
- Multiple views of the same content
- Drilldown is rarely used



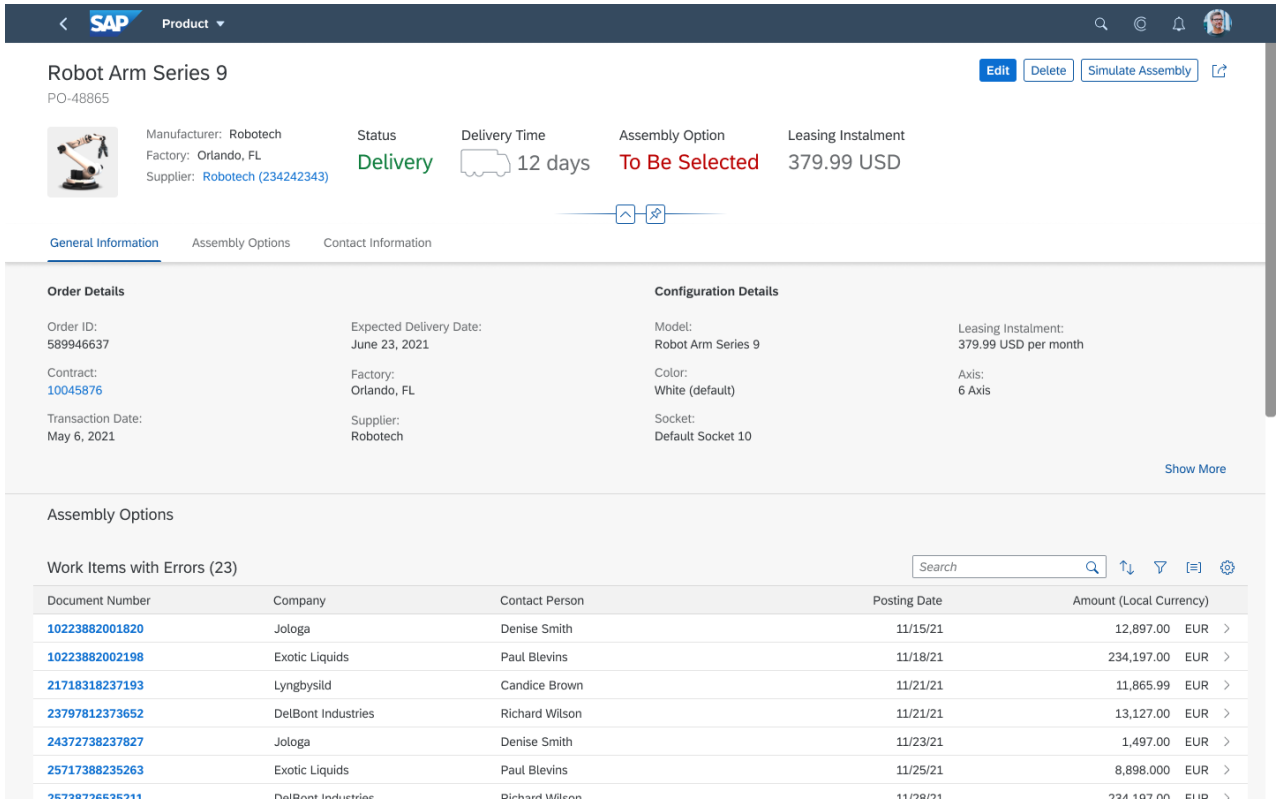
# Worklist Floorplan

- Collection of items a user needs to process
- Direct entry point for taking action on work items
- Similar to List Report.  If you have a large set of items and require searching, filtering, sorting, grouping – Use List Report



# Object Page Floorplan

- Display and categorize all relevant information about a single object



# Analytical List Page

- Offers a unique way to analyze data step by step from different perspectives
- Users need to extract key information to understand the current situation
- Large set of items that require searching, filtering, sorting, grouping, slicing and dicing

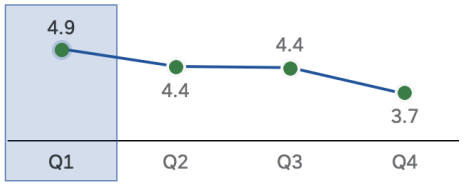
Standard*

RBQ 98.0M EUR | PBK 162,711 % | TQ 120K EA | GBP 116.6M EUR | TQ 120K EA | GAB

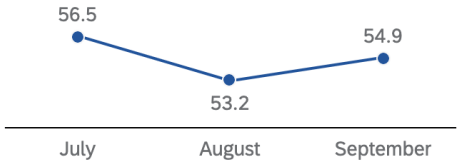
Global

*Revenue by Currency | M EUR (1)


Revenue by Quarter | M EUR

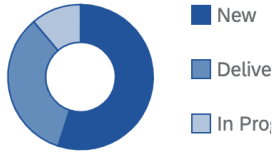



Net Amount by Delivery Month | M USD




Quantity by Date | EA


Net Product Price by Status | I



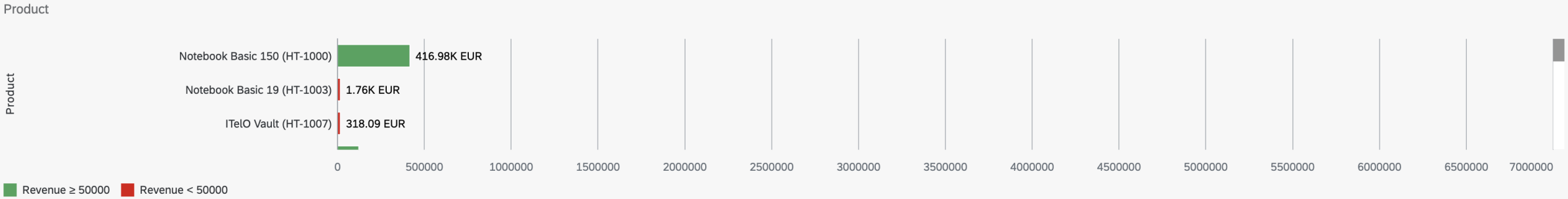

Adapt Filters (3) Go

## Sales Order Analysis

NBP 26.6K EUR | NAB 76.7M

### Sales Order Items

Save Current State | Navigation Button | Navigate | Details | View By

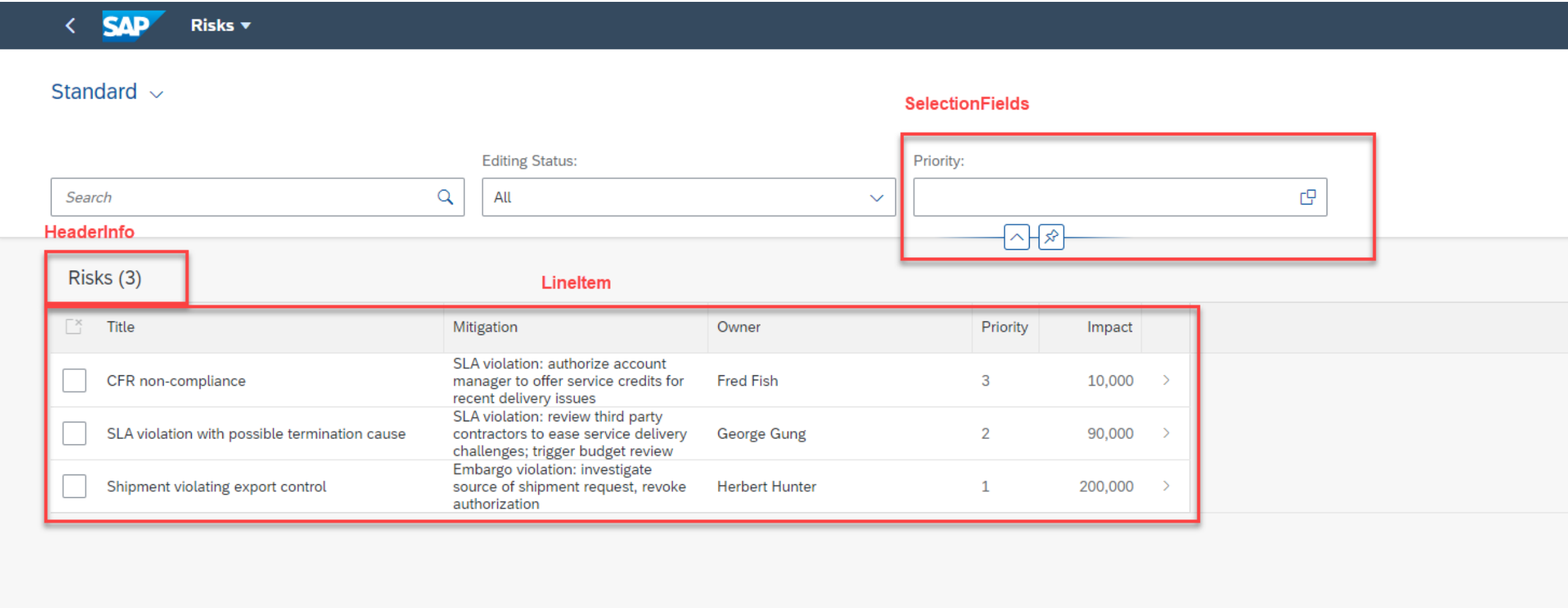


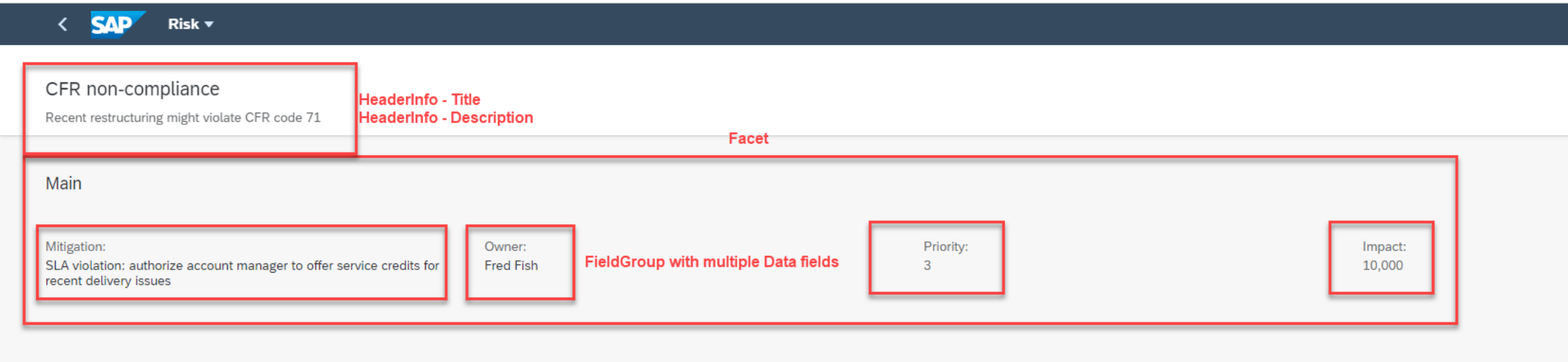

### Sales Order Items (9,263)

Save Current Time | Manage Sales Order | Navigate | Manage Order | Display Product

| Year | My Extended Colu... | Month | Product (IBN) | Customer | | | Sales Order | Item | Product |
|---|---|---|---|---|---|---|---|---|---|
| 2020 | data | January | Notebook Basic 150 | Japan Insurance Partner | Navigate Inline | Delete Order | 500008465 | 10 | Notebook Basic 150 (HT-1000 |
| 2020 | data | January | Cepat Tablet 10.5 | Japan Insurance Partner | Navigate Inline | Delete Order | 500008465 | 20 | Cepat Tablet 10.5 (HT-1257) |
| 2020 | data | January | Astro Laptop 1516 | Japan Insurance Partner | Navigate Inline | Delete Order | 500008465 | 30 | Astro Laptop 1516 (HT-1251) |
| 2020 | data | January | Astro Phone 6 | Japan Insurance Partner | Navigate Inline | Delete Order | 500008465 | 40 | Astro Phone 6 (HT-1252) |
| | data | | | | Navigate Inline | Delete Order | | | |

Determining | Chart/Table Context | Table Context | Navigation Chart | Submit | Complete Navigation | Update Order


Risks
Standard
SelectionFields
Editing Status:
Search
All
Priority:
HeaderInfo
Risks (3)
LineItem
Title
Mitigation
Owner
Priority
Impact
CFR non-compliance
SLA violation: authorize account manager to offer service credits for recent delivery issues
Fred Fish
3
10,000
SLA violation with possible termination cause
SLA violation: review third party contractors to ease service delivery challenges; trigger budget review
George Gung
2
90,000
Shipment violating export control
Embargo violation: investigate source of shipment request, revoke authorization
Herbert Hunter
1
200,000


Risk
CFR non-compliance
Recent restructuring might violate CFR code 71
HeaderInfo - Title
HeaderInfo - Description
Facet
Main
Mitigation:
SLA violation: authorize account manager to offer service credits for recent delivery issues
Owner:
Fred Fish
FieldGroup with multiple Data fields
Priority:
3
Impact:
10,000

# Steps involved

1. Initialize full-stack project – Completed (Unit 1, 2)
2. Create the tables – Data Modeling – Completed (Unit 1, 2)
3. Generic handlers – Out-of-the-box CRUD functionality – Completed (Unit 1, 2)
4. Basic UI - Completed
5. List Report layout - Completed
6. Custom event handling – Business logic
7. Support for external API
8. Connecting to Sandbox

# Step 6 – Custom Event Handling

No initial data assigned to Criticality field

Generic handler will return null for Criticality field

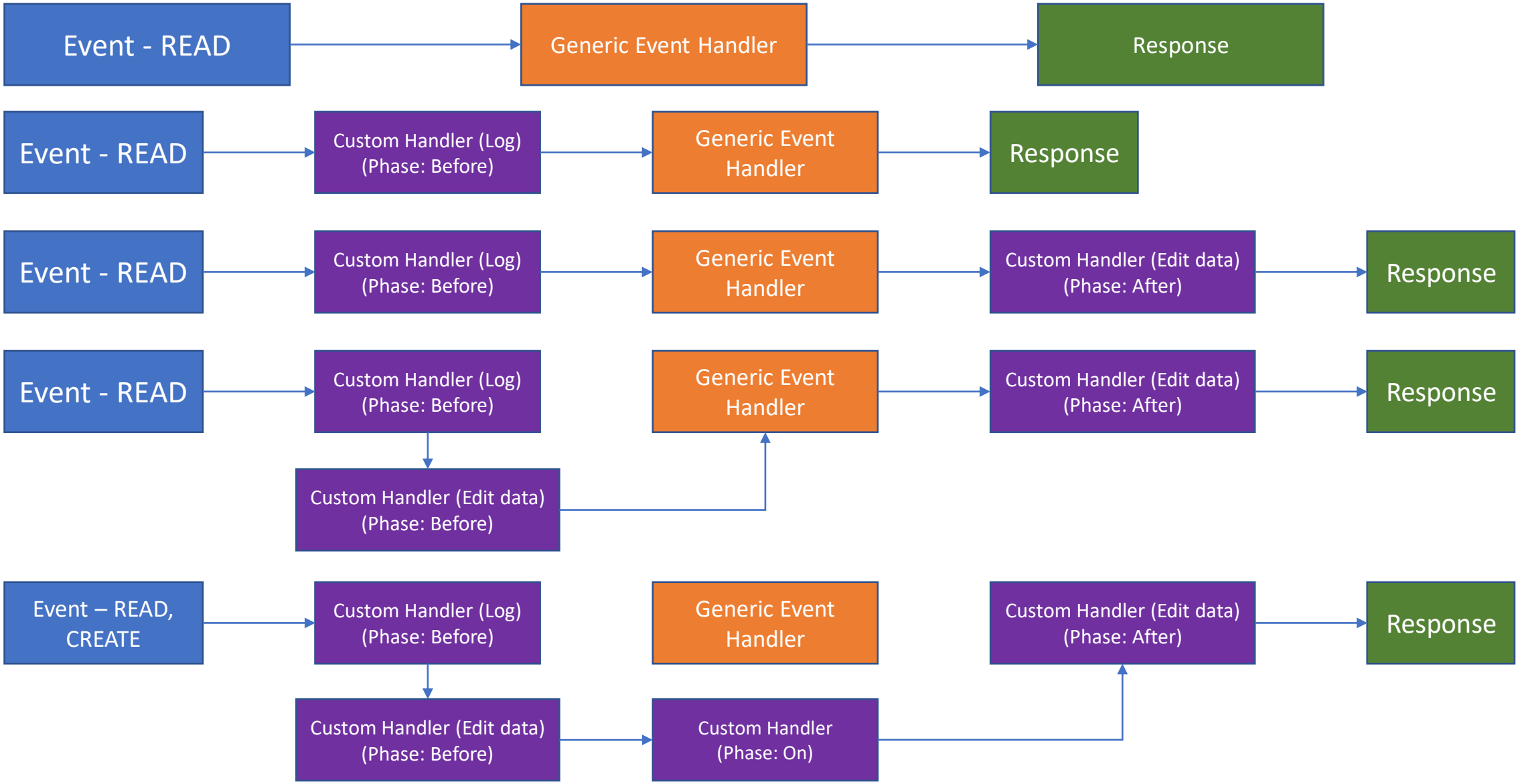
Business Logic:

Criticality = 1 – If value of impact field > 100000

Criticality = 2 – Otherwise

Custom Event Handling

Phase: after


Event - READ
Generic Event Handler
Response
Event - READ
Custom Handler (Log)
(Phase: Before)
Generic Event Handler
Response
Event - READ
Custom Handler (Log)
(Phase: Before)
Generic Event Handler
Custom Handler (Edit data)
(Phase: After)
Response
Event - READ
Custom Handler (Log)
(Phase: Before)
Generic Event Handler
Custom Handler (Edit data)
(Phase: After)
Response
Custom Handler (Edit data)
(Phase: Before)
Event – READ, CREATE
Custom Handler (Log)
(Phase: Before)
Generic Event Handler
Custom Handler (Edit data)
(Phase: After)
Response
Custom Handler (Edit data)
(Phase: Before)
Custom Handler
(Phase: On)

# Step 6 – Custom Event Handling

`git checkout 5_custom_event_handling (Use tab for branch name)`

New Files

- `risk-service.js`

Modified Files

- `annotations.cds`

Run application command

- `cds watch`

# Step 6 – Custom Event Handling

```
this.after("READ", Risks, (data) => {
    const risks = Array.isArray(data) ? data : [data];

    risks.forEach((risk) => {
        if (risk.impact >= 100000) {
            risk.criticality = 1;
        } else {
            risk.criticality = 2;
        }
    });
});
});
```

After Phase - data now contains all the Risks

Cover both reading a collection or a single entity

Loop through all the Risks

Set criticality value based on impact value

# Error Handling

**Guidelines:**

Fail loudly.  Do not hide errors and continue silently.

Log unexpected errors

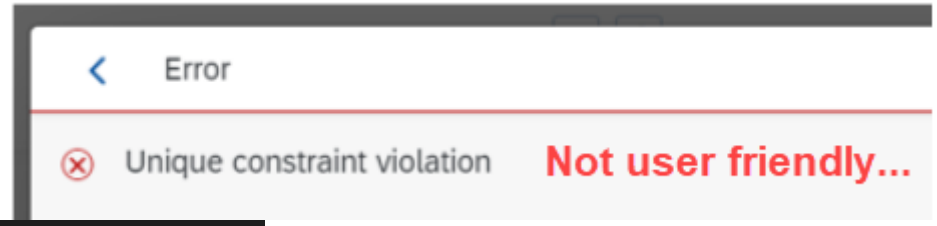
Don’t catch errors you can’t handle

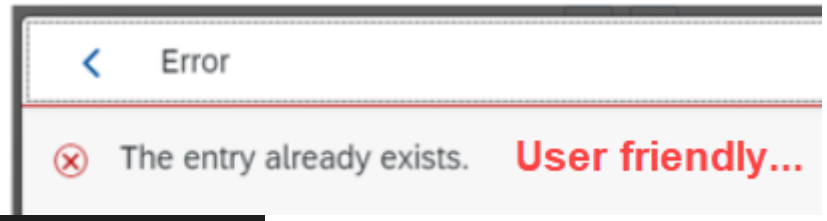
Use try / catch blocks only when necessary

# Error Handling

```
/**
 * Custom error handler
 *
 * throw a new error with: throw new Error('something bad happened');
 *
 **/
this.on("error", (err, req) => {
   switch (err.message) {
      case "UNIQUE_CONSTRAINT_VIOLATION":
         err.message = `The entry already exists.`;
         break;

      default:
         err.message = `An error occured. Please retry. Technical error message: ${err.message}`;
         break;
   }
});
```




Register an error handler in your service implementation

- Provide meaningful error message to end user

# Key Summary Points – Unit 3

Advantages of using SAP Fiori Elements

- Drive UX consistency
- Speed up development

Requirement:

- No unique functionality
- Fast development

Solution:  SAP Fiori Elements

# Key Summary Points – Unit 3

Standard floorplans of SAP Fiori Elements

- List Report
- Worklist
- Object Page
- Analytical List Page

# Key Summary Points – Unit 3

**List Report**

- Users need to find and act on items within a large set by searching, filtering, sorting and grouping
- Users need to work with multiple views of same content
- Drilldown is rarely used

# Key Summary Points – Unit 3

**Worklist**

- Users have numerous work items
- Users need to work with multiple views of same content
- Direct entry point for taking action on work items

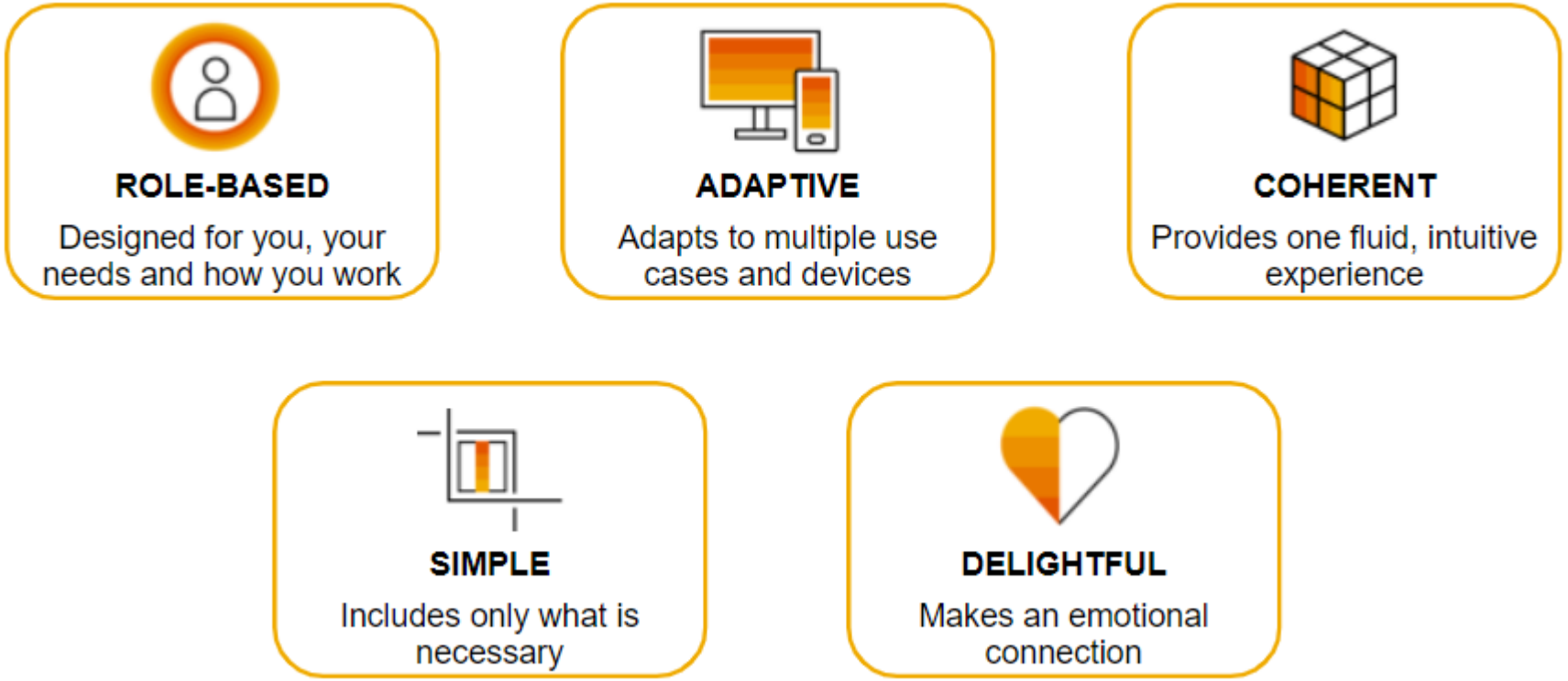
# Key Summary Points – Unit 3

**Analytical List Page**

- Users need to extract key information to understand current situation
- Users need to find and act on items within a large set by searching, filtering, sorting and grouping, drilling down, slicing and dicing

# Key Summary Points – Unit 3

## SAP Fiori Design Philosophy



# Key Summary Points – Unit 3

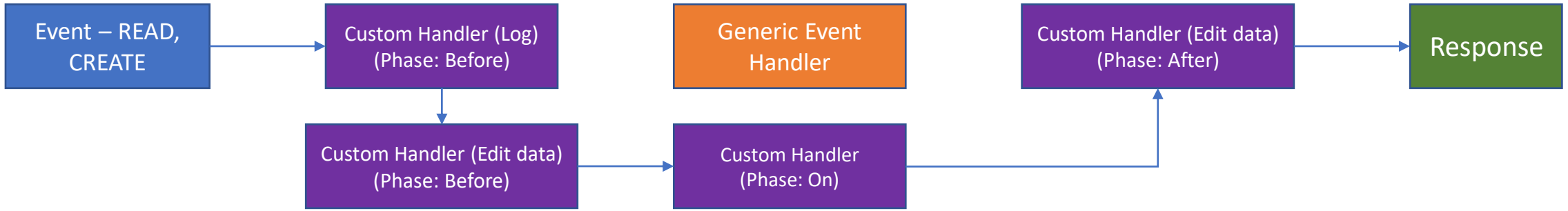
**Recommended folders for various artifacts**

**app** – UI artifacts

**db** – database artifacts

**srv** – Service related artifacts

# Key Summary Points – Unit 3




Multiple event handlers can be registered for each event phase

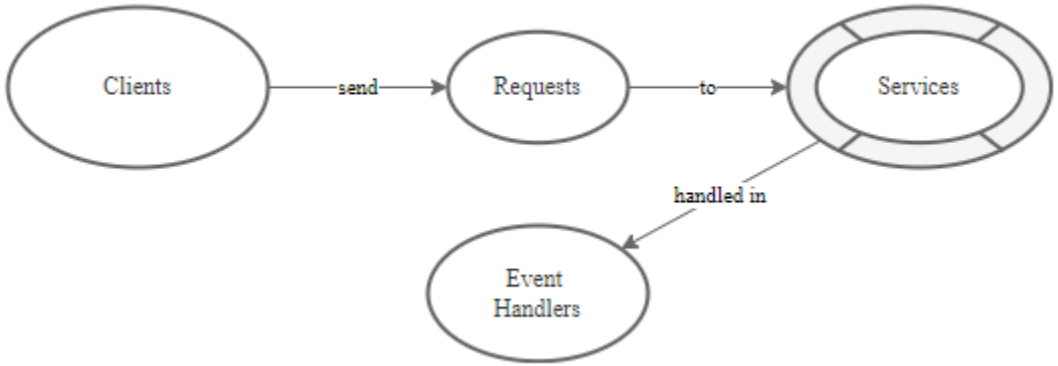
- The are executed in the order in which they appear

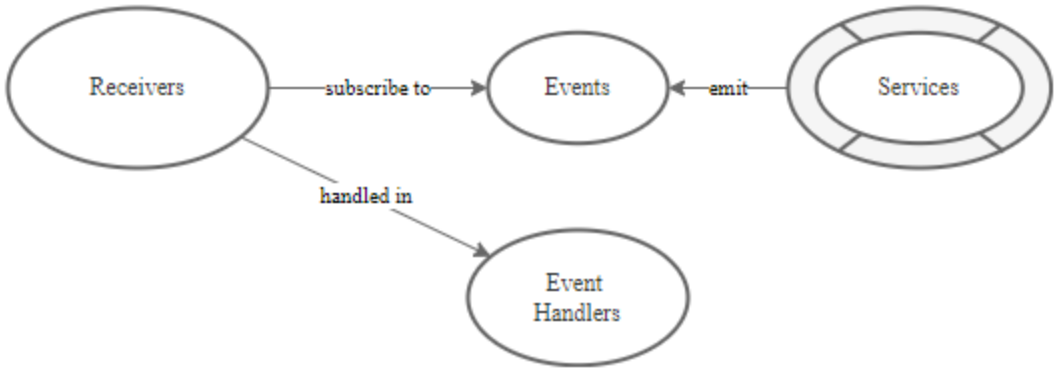
Single event handler can also handle multiple events

All services are event emitters

- Events can be sent to them – READ, CREATE etc.
- Events can be emitted by them
- Register event handlers with services to react to any event

# Key Summary Points – Unit 3



# Key Summary Points – Unit 3

### *req.**reply** (results)*

Stores the given `results` in `req.results`, which will then be sent back to the client, rendered in a protocol-specific way.

### *req.**reject** (code?, msg, target?, args?)*

Rejects the request with the given HTTP response code and single message. Additionally, `req.reject` throws an error based on the passed arguments. Hence, no additional code and handlers will be executed once `req.reject` has been invoked.

↳ *Arguments are the same as for `req.error`*

# Key Summary Points – Unit 3

### *req.**error, notify, info, warn** (code?, msg, target?, args?)*

Use these methods to collect messages or error and return them in the request response to the caller. The method variants reflect different severity levels, use them as follows:

*Variants*

| *Method* | *Collected in* | *Typical UI* | *Severity* |
|---|---|---|---|
| `req.notify` | `req.messages` | *Toasters* | *1* |
| `req.info` | `req.messages` | *Dialog* | *2* |
| `req.warn` | `req.messages` | *Dialog* | *3* |
| `req.error` | `req.error` | *Dialog* | *4* |

**Note:** messages with severity < 4 a collected and accessible in property `req.messages`, while error messages are collected in property `req.errors`. The latter allows to easily check, whether errors occurred with:

```
if (req.errors) //> get out somehow...
```

# Key Summary Points – Unit 3

Q10. What is the main idea behind SAP Fiori elements?

**Choose the correct answer.**

A Provide a framework and development tool kit for HTML 5.

B Define a role-based user experience (UX).

C Generate SAP Fiori apps at runtime from an existing OData service.

D Provide a showcase for the core principles of modern user interfaces (UI).

# Key Summary Points – Unit 3

```
this.after("READ", Risks, (data) => {
    const risks = Array.isArray(data) ? data : [data];

    risks.forEach((risk) => {
        if (risk.impact >= 100000) {
            risk.criticality = 1;
        } else {
            risk.criticality = 2;
        }
    });
});
```

# Key Summary Points – Unit 3

The key takeaways for programming errors are:

- **Fail loudly**: Do not hide errors and continue silently. Ensure to log unexpected errors correctly. Don't catch errors you can't handle.
- Don't develop in a defensive fashion. Focus on your business logic and only handle errors when you know they will occur. Use `try/catch` blocks only when necessary.

Never try to catch and handle unexpected errors, rejections of promises, and so on. If it is unexpected, you cannot handle it correctly. If you could, it would be expected (and should already be handled). Even if your apps should be stateless, you can never be 100% sure that a shared resource was not affected by the unexpected error. Therefore, you should never allow an app to continue running after such an event, especially for multi-tenant apps where there is a risk of information disclosure.

Following these guidelines will make your code shorter, clearer and simpler.