## **Bringing Fiori Apps into SAP BTP**

In the following sections I am trying to lay out the options that are there for SAP Standard Apps and Custom Build Apps to be brought into the BTP.

At the End I will give some rough estimates for the specific solution of the purchase requisition app for the different options.

My personal suggestion is to start trying Option 2 and if this fails go forward with Option 3 below.

## **Options for Bringing Standard Fiori Functions into SAP BTP**

To enable modern, scalable, and cloud-based access to standard Fiori functionality on SAP BTP, several architectural approaches can be considered. Below are the detailed explanations, benefits, and challenges for each option.

### **Option 1: Rebuilding in SAP Build Apps (Low-Code/No-Code)**

Recreate standard Fiori functionalities using **SAP Build Apps**, as a low-code/no-code environment, that enables visual development and rapid deployment of business applications.

**Benefits:**

* Rapid development with visual tools.
* Good integration with BTP services like SAP Mobile Services, Workflow, or Business Rules.
* Ideal for lightweight apps or extensions that don’t require deep custom logic.

**Challenges:**

* **Scalability Limitations:** Not well-suited for large or complex apps involving significant UI logic, multiple entities, or complex tabular data (like ALV tables).
* **Feature Gaps:** Build Apps lacks deep support for Fiori design paradigms (like Smart Controls or Smart Templates).
* **Data Binding Complexity:** Binding to OData services and managing large data sets can be cumbersome.
* **Developer Frustration:** Complex business logic or advanced UX patterns may need workarounds or custom JS widgets.

**Use Case Fit:** Lightweight apps, approval workflows, or simple forms; not ideal for applications with complex tables, dynamic filters, or editable lists.

### **Option 2: Migrating Existing Fiori Apps to SAP Build Work Zone (Advanced Edition)**

**Description:** Deploy existing SAP Standard and Custom Fiori apps from an on-premises system into the **SAP Build Work Zone**, giving users a centralized entry point for accessing apps hosted on BTP.

**Benefits:**

* Preserves existing app investments.
* Retains Fiori look and feel and application logic.
* Easily integrated into Launchpad services in Work Zone.
* Lower development effort if apps are already built and stable.

**Challenges:**

* **User Mapping Issues:** Typically, the BTP destination is configured with a **technical user**, meaning actions (like purchase requisitions) could appear to originate from the same SAP backend user unless mapped explicitly.
  + Requires identity propagation or user-substitution logic.
  + In SAP standard apps this usually isn’t prepared so the result would be that all purchase requisitions would be listed for each user, regardless of who created it. And also addresses, defaults etc. are shared among all users. The solution is unclear and typically app specific.
  + One Option to overcome this problem is to talk to SAP, if they would have a solution for it.

**Use Case Fit:** Good for organizations looking to expose existing functionality with minimal rework and still want to leverage a modern UX via Work Zone.

### **Option 3: Hybrid Rebuild Using BTP Deployed HTML5 + Reused Backend Services**

**Description:** Reuse backend OData or REST services from the on-premise system, but **rebuild the frontend** using modern UI5 or freestyle HTML5 in SAP BTP (deployed via the HTML5 Application Repository).

**Benefits:**

* Full control over the frontend; ability to create optimized UX.
* Can use modern JavaScript frameworks (UI5, React, etc.) depending on flexibility.
* Easy to modularize and scale compared to Build Apps.
* Better suited for complex apps (e.g., multi-table views, dynamic filters, advanced charts).

**Challenges:**

* **Higher Development Effort:** Requires full development lifecycle (coding, testing, DevOps).
* **Connectivity Setup:** Requires well-configured BTP destinations and cloud connector for accessing on-premise services.
* **User Propagation:** Needs handling of principal propagation or a substitute mechanism for user-specific operations. This problem is still there but it could be attacked by developing a custom strategy for storing user mappings like partners in purchase requisitions or similar options.

**Use Case Fit:** Best for medium-to-large apps or when redesigning for performance, UX, or mobile adaptation is desired.

### **Option 4: Use SAP Fiori Elements with CAP (Cloud Application Programming Model)**

**Description:** Create new applications in SAP BTP using **SAP Fiori Elements** and **CAP** (Node.js or Java). Reuse backend data via OData V4 services or replicate selected datasets into HANA Cloud.

**Benefits:**

* Declarative UI creation reduces coding.
* Full adherence to Fiori UX guidelines.
* Supports draft handling, list reports, object pages natively.
* Ideal for BTP-native development with future scalability.
* Built-in multitenancy and extension capabilities.

**Challenges:**

* Learning curve for CAP and CDS annotations.
* Requires backend data availability via APIs or replication.
* Not suitable for highly customized UIs (better for data-centric apps).

**Use Case Fit:** Ideal when building new cloud-native applications on BTP, especially if a data-driven app with list/detail/edit functionality is needed.

## **Summary Table**

| **Option** | **Best For** | **Challenges** | **Benefits** |
| --- | --- | --- | --- |
| Build Apps | Simple forms, quick POCs | Limited UI control, poor for complex apps | Rapid prototyping, visual dev |
| Work Zone Migration | Preserving investment | Identity mapping, network dependency | Fast enablement, Fiori consistency |
| HTML5 + Reused Services | Mid/large custom apps | Requires dev team, connectivity setup | Full UI flexibility, scalable |
| CAP + Fiori Elements | New data-centric apps | CDS & CAP knowledge, backend access | Declarative UI, future-proof |

## 

## **Effort Estimation Per Option**

| **Option** | **Estimated Effort** | **Effort Breakdown** | **Notes** |
| --- | --- | --- | --- |
| **1. SAP Build Apps** | **High** (20–30 person-days) | - Data modeling and bindings (10d)  - UI for PR & GR screens (10d)  - Testing and deployment (5–10d) | - Manual tabular entry of PR items is hard to build in Build Apps.  - No native support for Smart Tables, Approval UX.  - May require JS widgets or workaround logic. |
| **2. Build Work Zone with Existing Apps** | **Low–Medium** (5–10 person-days) | - Setup Work Zone (2d)  - Configure destinations, tiles, and roles (2–3d)  - Identity mapping & testing (3–5d) | - Leverages existing Fiori apps from S/4 or ECC.  - Must resolve user propagation if using technical users.  - If it is working this is the fastest solution, but could also be a dead end |
| **3. HTML5 App with Reused Services** | **Medium–High** (23–28 person-days) | - UI design (3d)  - Reuse and integrate OData services (10–15d)  - Identity handling & security (5d)  - QA & packaging (5d) | - Offers best flexibility but is full-stack dev.  - If services need enhancement, backend work increases. |
| **4. Fiori Elements + CAP (Cloud-Native)** | **Medium** (15–25 person-days) | - CAP project setup and CDS design (5–10d)  - Fiori Elements pages (5–7d)  - Connectivity and deployment (5–7d) | - Efficient for list/detail/edit flows like PR. - Not great if complex custom UI is needed. |