Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2021, 8(2):144-147

Research Article ISSN: 2394 - 658X

# A Review of Salesforce ISV Application Architecture Patterns for Scalable AppExchange Solutions

**Praveen Kotholliparambil Haridasan[1], Hemant Jawale[2]**

[1]praveenkhari@gmail.com
[2]heman.t.jawale@gmail.com

---

**ABSTRACT**

AppExchange is Salesforce's SaaS enterprise application marketplace. Salesforce customers can buy apps or solutions or services from the marketplace instead of building customizations themselves. The independent software vendors aka partners can adopt an application architecture pattern to build and scale their application that meets their customer requirements. In this paper, we will explore the various application architecture patterns available to the app vendors to architect and build Salesforce applications. We will also look at how to think of the AppExchange strategy to identify factors that will help determine which application architecture patterns are best suited to the customer needs.

**Keywords:** Application Architecture, Salesforce AppExchange, SaaS, Enterprise Application Marketplace

---

## INTRODUCTION

AppExchange was introduced by Salesforce in 2005. Since then, it has become the leading SaaS enterprise application marketplace. Certified consulting agents and independent software vendors (ISVs) aka Partners have their offerings listed on the Appexchange that help Salesforce customers extend their Salesforce functionality. You can find a variety of offerings like customizable apps and UI components. These offerings can solve a particular business challenge in an industry or in sales, service, or marketing. [1]

A Salesforce customer prospect will browse through the Appexchange to search through the app or solution or consulting service listings. A listing will have details around what functionality the app or service is providing, reviews from other customers, some screenshots or demo videos, and pricing. For an app vendor, Appexchange is a great platform for reaching a lot of Salesforce customers and running a business by building and selling applications.

## SALESFORCE APPEXCHANGE STRATEGY

For a partner looking to build and sell an application to customers, they need to first define a good strategy to ensure they can reach the scale and success to sustain their offering as a profitable business. These factors will help the partners determine which architecture pattern will work best for their customer requirements.

**Here are some things they need to think about:**

- **Product fit:** Is there a problem or a gap that Salesforce customers are experiencing today? If there is a demand for that, it is likely that an application that addresses this issue will have a good chance of success. Additionally, it should not compete with existing Salesforce product roadmaps or plans. Another success criterion for a solid product fit is that the application fits in the whitespace on Salesforce's product roadmap and augments the existing Salesforce functionality. This becomes a win-win scenario.
- **User personas:** It is critical to understand both the end user and the persona who would be purchasing the application. The incentives and difficulties these personas encounter must be accounted for while developing and designing the application.
- **Product roadmap:** The application should have a well defined roadmap of functionality and features. The input from focus groups or pilot customers comprising the aforementioned user personas would be valuable information to consider when ranking the features on the product roadmap.

- **Budget and resources:** There are important considerations to be taken when the project to plan, architect, and develop an application starts, such as how many resources are available to work on the application and how much money, time, and effort would be needed.
- **Technical considerations:** Understanding the many technical aspects requires an application architecture study. How much of the partner's well-developed technical solution on their own technology stack can be integrated with Salesforce's technology stack, for instance? Can the partner create the functionality on Salesforce's platform from scratch if it isn't already built out? A few key choices have the power to make or break an application. Understanding the different application architecture alternatives can help the partner make that selection with confidence.

## APPLICATION ARCHITECTURE PATTERNS

The SaaS enterprise applications that partners develop for Salesforce are used by multiple customers. Partners have to architect the application in such a way that it scales for all kinds of customers. Reusability, scale, and robustness become key factors when they want to architect such applications. The work and through that goes into selecting the best application architecture option makes it relatively easier to distribute at scale and maintain for future innovations and bug fixes.

There are 3 main application architecture pattern options when we talk about how partners architect their Salesforce application.

### 1. Salesforce Native Application

This option is ideal if partners are trying to architect an app solution with no or minimal existing functionality or a technology stack. This means that they are most likely building from scratch. Salesforce platform [2] provides all the tools and technology for a partner to create an application natively on Salesforce. This means there are no integration points to external systems. Salesforce conducts 3 releases per year [3] providing the latest innovation and updates on their platform. They also ship a lot of out of the box capabilities and functionality. Native Salesforce applications can not only leverage these innovations and out of the box capabilities, but also create customizations that meet customer requirements. For a customer who is buying and installing a native applications, the app will sit on top of the customer's Salesforce org instance natively.

The Figure1 below represents how a partner packaged app will embed itself in a customer's Salesforce organization instance. The customer will have the out of the box Salesforce capabilities and can additionally have their own customizations. The partner application will sit along those capabilities and provide the functionality the customers need that they don't want to build themselves. The partner applications can consist of various metadata including custom objects, UI (User Interface) components like Lightning Web Components (LWCs), and custom flows or apex code for automations.

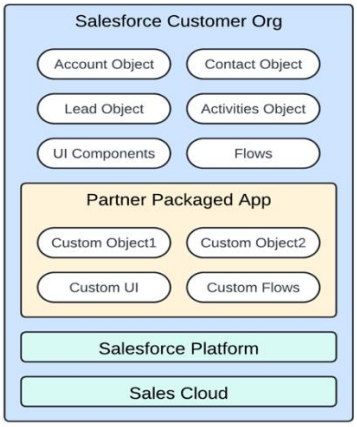


Figure1: Salesforce Native Application in a Salesforce customer org

### 2. Hybrid Application

Hybrid application, as the name suggests, uses a hybrid approach for which technology stack partners can use to build their application processes on. If a partner has their existing processes running on their own technology stack outside of Salesforce, they can reuse that functionality by creating an API integration with Salesforce. While doing so, they can still build some functionality on Salesforce to ensure the customer user experience feels much more Salesforce native to them. This option is best suited when the end users of the application like to spend time on Salesforce and they don't want to navigate away from that to another system.

Refer to Figure2 below as an example. If the partner has data stored in an AWS S3 bucket that is part of their technology stack, they can create an API integration where they sync data from their endpoint into Salesforce custom objects that are part of the packaged application. They can also build UI Lightning Web Components to

show that information in Salesforce to the user. Once the user takes action on that data in Salesforce, the application can sync those updates back to S3 bucket in a batch fashion. This way the application can perform a bi-directional data integration between the two systems.

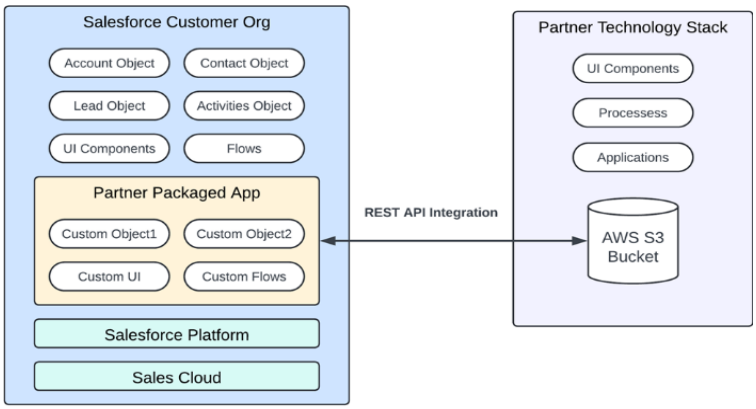


Figure2: Salesforce Hybrid Application in a Salesforce customer org

**3. API Only Application**

This type of application architecture pattern is used when partners want to completely rely on all of the capabilities and functionalities of their existing stack and not build new UI or processes on the Salesforce platform. This pattern is best when partners know that their end users will spend time on their own system rather than Salesforce. This pattern relies heavily on API integrations. There is a feature in Salesforce called the Connected Apps [4] that is recommended for a partner to use. Partners can create this connected app as part of their application that provides the integration setting controls for the Salesforce admin to set up their application integration with Salesforce. The linked connected app lets the Salesforce admin know the details about the external application's system and the authentication mechanism it utilizes, which can include OAuth, OpenID Connect, and SAML. After that, Salesforce provides policies that specify access limits, like the expiration date of the app's access, and allow the external app access to its data. Salesforce is also able to audit the use of the linked apps.

Figure3 below is an example of how an API-only application could be architected. Here we see the connected app as a component of the partner package application that provides the setting controls to enable and maintain the REST API integration to the partner end point service.

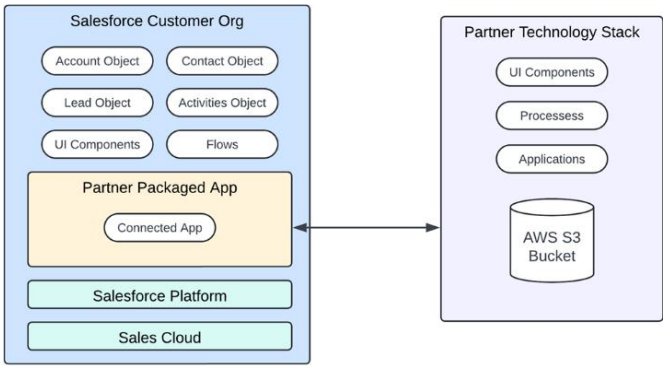


Figure3: Salesforce API-Only Application in a Salesforce customer org

## INTEGRATION PATTERNS FOR SALESFORCE APPS

We talked about the 3 app integration patterns above. When it comes to the hybrid approach and the API only approach, there are several ways to go about the actual API integration. Salesforce has well documented information on the kinds of integration patterns available and which ones are best suited to what scenario [5].

Data integration using REST APIs is the most common approach the apps would take or need. The REST API developer guide [6] that Salesforce has published is a good start to understand how partners can start integrating the data from the custom app objects that they will create with their endpoint service data and vice versa.

There are few things partners need to think of when defining their app integration pattern.

**1. Source of Truth:** Where will be the source of truth for their data? Will Salesforce or their own database/data lake be the main source of truth where the data first flows into?

**2. Data Sync Frequency**: Does the data integration needs to happen in real-time? Or can the application wait for an amount of time for all the data updates and sync those in a batch fashion? Doing it in batches saves a lot of resources and API consumption.
**3. Data Volume:** How much data is moving? If the data is in the higher millions, the Salesforce application might need a middleware to offload some processing or the integration patterns have to be optimized to support that level of data sync that provides consistent performance.

## LIMITATIONS AND CHALLENGES

Every application is unique. The customer problems the partners are trying to solve and the functionality they provide differs from one another. This means the application has to be architected in a way that will adhere to the limitations of both their own technology stack and Salesforce platform. Performance could also degrade over time if the integration pattern is not optimized for scale. These limitations have to be kept in mind when architecting an application.

## CONCLUSION

Salesforce AppExchange is a unique value proposition for a company or an individual to become an app vendor. Salesforce provides the platform and capabilities to partners to architect their applications that suit their technology capabilities and preferences.

- Native applications are best suited if partners do not own a technology stack and want to leverage Salesforce's product innovation. They can meet their customers where they are, in this case, on Salesforce.
- Partners can take a hybrid application architecture approach to leverage the best of their existing technology and processes while also investing in Salesforce. This allows their end users to stay in their preferred work of flow which would be Salesforce.
- API-Only approach is best suited if end users prefer to work in a partner's own system and need a lightweight integration to Salesforce.

## REFERENCES

[1]. AppExchange [Online]. https://appexchange.salesforce.com/
[2]. Salesforce Platform [Online]. https://trailhead.salesforce.com/content/learn/modules/starting_force_com
[3]. Keep Up with Salesforce Releases [Online]. https://help.salesforce.com/s/articleView?id=sfdo.SFDO_Keep_Up_SF_Rels.htm&type=5
[4]. Connected App [Online]. https://developer.salesforce.com/docs/platform/einstein-for-devs/guide/einstein-overview.html
[5]. Integration Patterns Overview [Online]. https://developer.salesforce.com/docs/atlas.en-us.integration_patterns_and_practices.meta/integration_patterns_and_practices/integ_pat_intro_overview.htm
[6]. REST API Developer Guide [Online]. https://developer.salesforce.com/docs/atlas.en-us.api_rest.meta/api_rest/intro_what_is_rest_api.htm