Implementing an API-First Design Methodology

(/users/3617470/dnrb.html) by David Brown (/users/3617470/dnrb.html) </>CORE · Apr. 09, 19 · Integration Zone (/enterprise-integration-training-tools-news) · Opinion

Like (10) Comment (0) Save Tweet

What Is an API-First Design?

Applying an API-first approach means designing an API so that it has consistency, as well as adaptability, regardless of what development projects it's applied to. Using an API description language, such as OpenAPI, for your API is key, as it helps establish a contract for how your API communicates with other programs — even when the language behind these systems is unknown. Of course, APIs don't exist in a vacuum, so collaborating and planning with organizational stakeholders is equally important to the design process, which begins long before any code is actually written. In short, implementing an API-first strategy requires you to think beyond the here and now of the design process.

Why API-First Design Is Important

Traditionally, the software design process begins once a problem is identified and someone realizes that a new program may be able to provide the necessary solution. Once the broad strokes are established, the next step usually involves digging a bit deeper to assess any possible use cases and forms of interaction that may be required of the system being created. Of course, all of this must be done in consideration of the user experience (UX) because having a system that works isn't enough; it also needs to be user-friendly for the developers and consumers who are going to interact with it, be it on the backend or the front end.

Sometimes the problem is even simpler — you've got an internal application with data that you'd like to make accessible to other systems via an API. However, this system has its own logic that informs how it behaves and interacts with others. Therefore, any API that you create will be built around these idiosyncrasies and constraints. Once your internal application is connected as an API, developers interact and attach with it dependent on the system's internal structure. This may be fine if that's what you intend, but issues can arise if you later decide that you want or even need to make changes to your internal model. Designing with an API-first approach means building an API that is more than just a byproduct of an internal system.

Developers should be able to quickly and easily understand how your API works and integrates with other applications. Only then can they can write the kind the elegant code that will allow it to efficiently interact with other systems. When an API is correctly implemented, it is both backward compatible and future proof; each of which is reflected in everything from the scope of the application, its schema, and parameters, in addition to organization and industry best practices.

Advantages of an API-First Design Approach

- **Development teams can work in parallel**
  - Having a contract allows developers to work on different sides of an API simultaneously, without the time and cost associated with waiting for application updates. As a result, developers can mock APIs and effectively test any relevant dependencies all based on the pre-established contract. An increase in productivity and efficiency is also likely to occur as a result.

- **Lower cost of developing applications**
  - The reusability of an API-first design approach allows code to be recycled from project to project so that development teams always have a baseline architecture with which they can work. By eliminating the need to code from scratch, any associated time and financial costs are similarly reduced. In addition, developers are able to troubleshoot API issues more quickly and often without any code, allowing for smoother integration with other applications.

- **Increase speed to market**
  - Automation-backed "discoverable" APIs give even novice developers the ability to quickly and easily interact with API documentation. Indeed, much of the API construction process can be automated with tools that have a visual API designer, can import API definition files and automatically generate services to consume the API, generate documentation, or even mock an API's response. Such tools dramatically reduce development time and increase the speed at which your products come to market.
  - Most importantly, with an API-first approach, you still have the flexibility to add new application product features without having to redesign the API architecture. This is a distinct advantage in the modern landscape of agile application development whereby application updates are expected to be released quickly in response to end-user demand.

- **Improved developer experience**
  - Developers interact with APIs on a daily basis so creating an API that provides a positive developer experience (DX) is essential. An API first design

approach yields an API that is well-designed, as well as, well-documented and consistent to its core. Developers can use the API to more easily integrate with other applications, quickly troubleshoot any issues that arise and even onboard other developers in less time, thanks to a less formidable learning curve.

- **Reduced risk of failure**
    - For most businesses, APIs are an integral part of the operational landscape, touching everything from marketing and sales to the consumer-facing apps that represent their brand. An API failure in any part could be devastating. But with an API first approach, the possibility for error is greatly reduced due to the inherent reliability, stability, and consistency of the design and implementation.

API Design in the Context of Application Integration

APIs provide a uniform structure for communication between systems, be they new or legacy, allowing for the transmission of data its transformation. But API implementation isn't just an IT issue. An API must be designed with respect to the goals and objectives of the organization at large. With an API strategy at the forefront of your design process, application integration and interoperability between systems can be improved and, in many cases, optimized. However, you still retain the option to make changes if needed.

What Are the Characteristics of a Well-Designed API?

The most effective API designs include the following attributes:

- **Easy to read and work with:** A well-thought-out API is easy to read, comprehensively documented, and peppered with straightforward examples that support developer use. Resources and parameters are intuitive and therefore easy to learn and remember.
- **Hard to misuse:** Implementing and integrating with the API is easy and linear, making it difficult for developers to misuse it — even when writing coding. It also contains informative feedback while simultaneously permitting a fair degree of flexibility on the developer's end.

flexibility on the developer's end.

- **Complete and concise:** The API is comprehensive enough to allow developers to incrementally build full applications on top of it using the data it communicates, well into the foreseeable future. This longevity and flexibility represent the type of design ideas to which every API creator should aspire.

Adopt an API-First Design Methodology and Supercharge Adoption of Your APIs

An API-first approach can have a powerful effect on the adoption and consumption of your APIs. This is especially true if your organization's API goals include a high-adoption and retention rate, or providing a great developer experience. The most effective API design provides a seamless consumer experience with easy-to-understand resources and clear value propositions, allowing for quick integration and usage. Likewise, with a reduced learning curve, the API is significantly more likely to enjoy reusability and continued engagement by developers.

Topics: API, INTEGRATION, APPLICATION DEVELOPMENT, API-FIRST DESIGN

Integration Partner Resources

Zone Leader Program (/pages/zoneleader)
Become a Contributor (/pages/contribute)
Visit the Writers' Zone (/writers-zone)

(/) (/users/login.html) (/search)

REFCARDZ (/refcardz) RESEARCH (/research) WEBINARS (/webinars) ZONES

LEGAL
Terms of Service (/pages/tos)
Privacy Policy (/pages/privacy)

CONTACT US
600 Park Offices Drive
Suite 150
Research Triangle Park, NC 27709
support@dzone.com (mailto:support@dzone.com)
+1 (919) 678-0300 (tel:+19196780300)

Let's be friends:

in (https://www.linkedin.com/company/devada-

(/pages/feeds) (https://twitter.com/DZoneInc) (https://www.facebook.com/DZoneInc)

DZone.com is powered by AnswerHub logo (https://devada.com/answerhub/)