Product Development Solutions

Big Nerd Ranch eBook

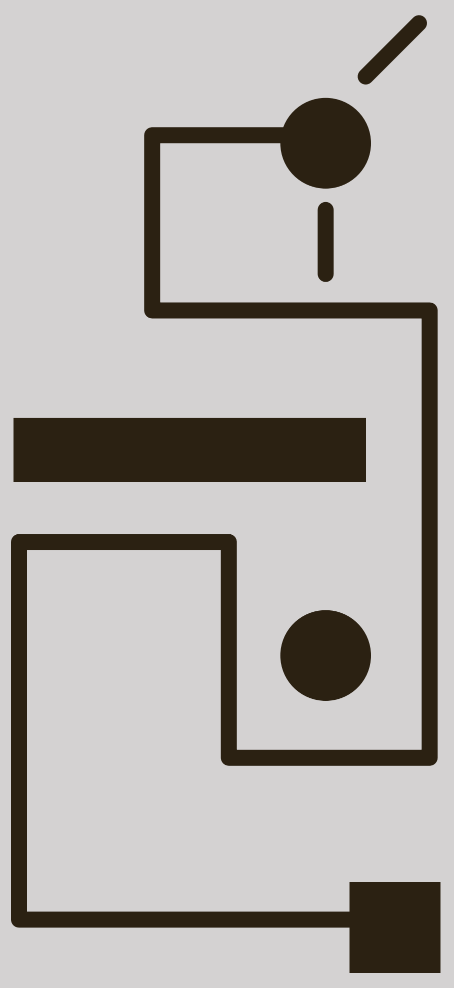
API-First Design

How some of the most important work is done before the first line of code is even written

Big Nerd Ranch

Executive Summary

- Backends can suffer quality problems leading to hidden costs. These can lead to slow delivery and errors, impacting the bottom line.
- Low quality backends stem from building accidentally rather than intentionally. Building backend features as needed seems productive but leads to inconsistency.
- Your API is a first-class citizen, a product that needs an owner. High-quality products don't happen accidentally. Choosing an owner cements the focus.
- Design the API first, then generate docs, interfaces, and mocked backends. Machine-readable formats pay off by generating other resources, keeping them in sync.
- Build frontends and backends in parallel to get to market quicker. You can develop multiple frontends in parallel without being blocked by the backend.



API-First Design is an approach to building web services that are high in quality and avoid the common hidden costs of web services.

API-First Design

Before we can talk about what API-First Design means, we need to define a few terms as we use them in this paper.

Web Service: a system providing data over HTTP in a machine-readable format, such as JSON, XML, GraphQL, or Protocol Buffers.
API, or **API Contract:** the publicly-visible interface of a web service: which endpoints are available, what authentication it uses, the format of input and output data.

These terms are often used interchangeably, and it's not wrong to do so. But it's valuable to make this distinction. In fact, as we'll see, making this distinction is precisely the core of API-First Design.

To best understand the importance of API-First Design, it's worth examining the cost of starting a project without putting your API front and center.

The Hidden Cost of Web Services

When it comes to organizations' software systems, web services often suffer from low quality which can have significant financial impacts if left unchecked. The problems that frequently come up include:

Inconsistent Structure: paths, field names, data types, response codes, and error messages can vary endpoint to endpoint for no discernible reason.
Documentation is Missing or Outdated: there isn't an easy way to see what endpoints exist and what data they require. If documentation is out of date, it's misleading and may cause more problems.

Undefined Behavior: there aren't ways to tell all the possible data types or errors that may be returned from an endpoint.
Undescriptive Errors: errors may be very general without providing details necessary to discover what the problem was. Internal errors from different libraries may be passed along without making them consistent or giving necessary context.

The root cause of poor quality web services is that they are accidental instead of intentional. Nobody intends to build a poor quality web service but when they evolve without being intentionally guided, they end up with inconsistencies:

- Web service endpoints are added to pre-existing server rendered web applications.
- Features are only added to the web service at the time they are needed for a client.
- Because functionality is built ad-hoc upon request and without a plan, it is often built as quickly as possible, without an overall strategy.
- Internal data structures and errors are exposed directly because it's the quickest way, despite higher long-term costs.

By contrast, what does it look like when a web service is built with intentionality? What does it look like when we achieve the goal of a high-quality web service?

Consistent: parts that function the same have the same structure and all differences are there for a reason.
Reusable: built in a client-independent way to allow multiple current and future clients to access it.
Well-Documented, ideally **Self-Documenting:** so developers can easily understand how to get the full benefit of the web service without missing or misunderstanding features.
Easy for Developers to Use, to avoid unnecessary development costs.

This would be great, but how can we get there? How can we approach building our web services to end up with high quality?

"The root cause of poor quality web services is that they are accidental instead of intentional."

So what is the API-First Design process?

There are five main steps to the API-First Design process.

1. Treat the API of your web service as a first-class citizen.
2. Appoint product ownership for the API to keep the focus on API quality high.
3. Design the API contract before building the web service in code.
4. Automatically generate documentation, client/server stubs, and mocked backends from the API contract.
5. Build the frontends and backend in parallel based on the API contract.

Let’s look at these steps in detail.

1. Make the API a First-Class Citizen
First, your organization needs to decide whether or not you believe that the cost of the problems in your web service API is great enough to justify the cost of an API-First Design process to improve it. Like any other process, API-First Design isn’t free. It takes time to learn, is an ongoing effort, and will probably require one or more people to staff the project. It’s only worth it if it’s fixing problems that are costing you money.

To help make this decision, think through examples and trends of bugs and development delays in your systems, and see which of them are attributable to backend design problems. The more of these you can find, the more likely API-First Design will pay off for your organization.

Another factor to consider is how important it is for your company to adapt to new client platforms in the future. Staying competitive in your industry might mean you need to be able to reach customers where they want to be reached, whether via voice assistants, chatbots, or Augmented Reality apps. If your company needs to be able to quickly adapt to new platforms, high-quality backends are essential to make that happen, and API-First Design can help enable this.

2. Appoint API Product Ownership
If you decide that API-First Design is cost-effective for your organization, your next step is to appoint product ownership for your API. Your products have product owners, product managers and/or project managers to advocate for them, plan for them, and guide them toward success. Your APIs need this leadership as well.

"Appointing an API product owner or similar role ensures a focus on the quality of the API contract."

It may be a new concept to think of APIs having product ownership. But if your APIs don't have this leadership, they will be changed in an ad-hoc way, leading to poor quality. Appointing an API product owner or similar role ensures a focus on the quality of the API contract.

An API product owner is not a developer or architecture role. Some technical experience can be helpful for the position to better understand foundational concepts of web services. But this person is advocating for your process and getting different stakeholders talking–technical and non-technical.

3. Design the API Before Implementing
Once you have product ownership to drive the API-First Design process, your next step is to design the API before building it. Remember that the API is different than the web service implementation. Resist the temptation to start writing code!

Note that this process can be applied whether you're building an entire web service from scratch, adding on to an existing web service, or evolving it to a new version. In all of these cases, you are designing a new API for some or all of your web service and you can apply API-First principles.

To design the API, involve a wide range of stakeholders in the API definition process. This should include frontend developers, backend developers, UI/UX designers, and business representatives. All of these roles have a stake in what functionality your API makes available, how it's structured and accessed.

Choose a tool that you will use for your API design process, and make sure that that tool is not code. It could be as simple as a white board, or a basic drawing program. A more structured diagramming tool might work, but make sure all the stakeholders can easily understand what you are designing and participate in making changes.

As your design solidifies, translate the API contract to a machine-readable format, such as OpenAPI or GraphQL Schema. This format will have major payoffs in the next step.

4. Generate Docs, Interfaces, and Mocks
A lot of time, designing and creating documentation doesn't feel very productive, as the documents are stored somewhere and never referenced again. That's why it's important to store your API contract in a machine-readable format like OpenAPI or GraphQL Schema. These formats allow automatically generating a number of extremely useful artifacts that will help your development process go faster:

Documentation: you can set up a pipeline to automatically generate a documentation web site whenever the API contract changes, and automatically publish it somewhere that's easily accessible by all stakeholders. This ensures that everyone is aware of the functionality. Tooling even exists to allow users of this documentation to make requests to mocked or development web services directly from the documentation web pages, to try them out and see how they work.

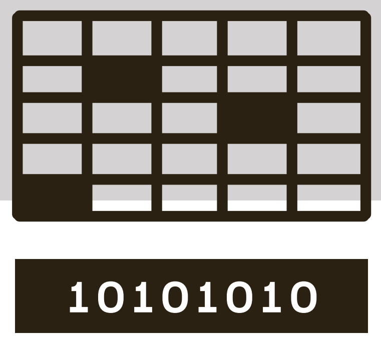
Client connection code and server interface stubs: these can be generated to save you time writing the connectivity code on the client and server side. This also has the added benefit of ensuring developers don't misinterpret the documentation and have to correct it later.

Mocked backend: existing tooling allows you to generate a backend with “mock data” that is hard coded, or follows very simple logic. This provides you with a running version of the API that frontend developers can build against right away. They can have a high degree of confidence that as the functionality of the backend is built out, it will match the mocked backend since both are derived from the same API contract.

5. Build Frontend and Backend in Parallel
With automatically generated documentation, interface stubs, and a mocked backend, you are set for frontend and backend development to proceed in parallel.

Working in parallel is normally a risky proposition. Even if your teams think they have a mutual understanding of how the backend will work, without the rigor of a thorough API contract, you are likely going to need significant changes to the backend. This results in rework across all your frontend apps that have already been built based on the old assumptions.

"This process helps ensure your APIs are high-quality, so you can avoid hidden costs."



With an API contract produced by an API-First Design process, you avoid a situation where frontend and backend developers interpret a loose specification differently. Also, all your stakeholders have already thought through the API in detail and come to a consensus. This makes the number of changes needed much less frequent. That being said, it’s likely that changes will still be necessary, but when the need arises, you will have a clear starting point of the API contract to discuss what needs to change and why, and you have a group of stakeholders who are already practiced at working together.

As your backend is built, tooling can test it against the API contract to ensure it conforms. This can catch errors earlier, and avoid a situation where a frontend application gets an error and it’s not clear if the mistake is on the frontend or backend side.

How API-First Design Helps

Now that we've looked at the steps of API-First Design in detail, let's review them again:

1. Treat the API of your web service as a first-class citizen.
2. Appoint product ownership for the API to keep the focus on API quality high.
3. Design the API before building the web service in code.
4. Automatically generate documentation, client/server stubs, and mocked backends from these design files.
5. Build the frontends and backend in parallel based on the API design.

This process helps ensure your APIs are high-quality, so you can avoid the hidden costs of low-quality web services that can hinder your organization. Let's look at how:

Inconsistent Implementation: Designing your API gives you an opportunity for interface thinking, to avoid unintentional interface inconsistencies and implementation details leaking through. API design tools provide ways to reuse data structures, so that it's less work to be consistent and more work to be inconsistent.
Documentation Missing or Outdated: Because the API contract is designed apart from implementation, any artifacts of the design process serve as documentation. Documentation in various other formats can be automatically generated from the API contract. Code stubs can also be automatically generated from the API design, and the backend can be automatically tested against the API contract to ensure it conforms.
Undefined Behavior: the API contract is a way to specify the behavior of the backend.
Undescriptive Error Messages: API design tool functionality for error messages serves as a reminder to design for them. Designing the error messages makes it more likely they will be consistent instead of whatever errors the implementing code happens to throw.

Resources

To learn more about API-First Design, here are a few helpful resources:

Put APIs at the Center of Your Digital Business Platform - Gartner

Understanding the API-First Approach to Building Products - Swagger

Three Principles of API First Design - Adobe Tech Blog

Understanding API First Design - ProgrammableWeb

About Big Nerd Ranch

Since 2001, Big Nerd Ranch has worked to bring brilliance to life. We achieve this by establishing true partnerships with our clients and by providing a team of engineers, designers, and project strategists to guide your digital product from discovery to launch. These same developers and designers can join your company's team to take a project across the finish line or train them on the latest and greatest technologies. We're proud to work with startups and Fortune 100 companies building authentically useful applications and transforming vision into advantage. And, as a winner of the 2019 AJC Work/Life Balance Award, we work as hard on our culture as we do on your projects. Learn more at www.bignerdranch.com and email marketing@bignerdranch.com if you'd like more information on this press release.