STRV

STARTUP FOUNDER TECH GUIDE

How to Choose the Right Tools to Succeed in 2020

WWW.STRV.COM

Intro **1**

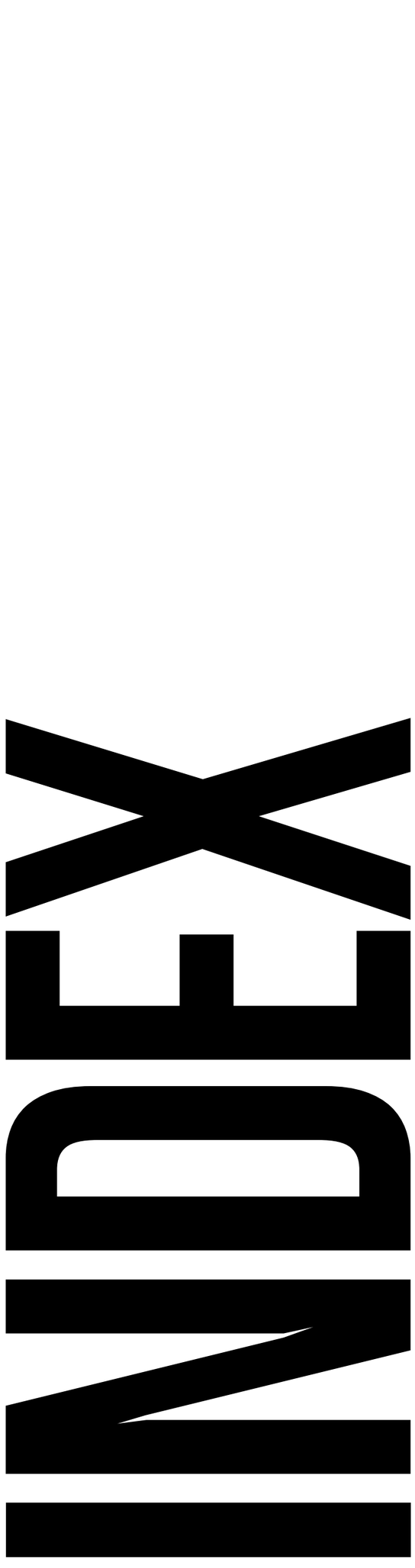
Mobile Development **4**
Native 6
React Native 9
Flutter 11
Progressive Web Apps (PWAs) 13

Backend Development **16**
Javascript/Node.js (Typescript) 18
Python 20
C#/.NET 22
Golang 24
PHP 26
Java 28
Databases 29
Deployment & Hosting 33

Frontend Development **41**

Quality Assurance **48**

Product & Project Management **52**



Introduction

The tech world evolves at a speed difficult to grasp for those not expert in the field. There are innumerable options in terms of frameworks, languages and tools when it comes to choosing the right path for your digital product. This can be overwhelming for many, but it's great for engineers who are always looking for better solutions to complex problems. Just as you want the best product possible, your engineers want to find the best way to get there.

Determining what works for you is a process. It should be the result of thoughtful consideration and requires experience. What does your product need? How is it meant to transform the industry? What has worked in similar scenarios in the past? An awareness of both the advantages and pitfalls of all options allows for comparing and testing alternatives.

The purpose of this ebook is to share our 16 years of observations with building and scaling digital products. We aim to provide an objective overview and high-level comparison of the various options available, helping you bring your product to life. And, more importantly, helping you ensure that it stands out amongst the ever-growing competition.

Making the right choice is not always simple. We want to provide guidance in making decisions that make sense from a technical and business perspective. With input from all of our platforms' senior experts, we hope we can help make things just a little bit easier.

— Lubo Smid, Co-Founder & CEO

MOBILE

Jan Schwarz
iOS Platform Lead

Petr Nohejl
Android Platform Lead

Jan Kaltoun
iOS Platform Lead

Juraj Kuliska
Android Platform Lead

Jan Pacek
iOS Platform Lead

Introduction

Choosing the right framework for building a new app can make or break a successful release and the future of your business. In the past, engineers had little to choose from. But as the market and competition grew, so did the offer of development frameworks. That's great news. The tricky part now is figuring out what's best for you. From a business perspective, there are two primary pillars on which your decision should stand: complexity and budget.

Developing a cross-platform app is usually cheaper and faster than building an app for both major mobile operating systems (Android and iOS). But it's not that simple. In terms of complexity, non-native development comes with complex implementation of platform-specific features, limited functionality, lower performance, worse user experience and tough, sometimes impossible hardware integrations. So is it worth it? Depends on what you're trying to do, who you're doing it for and how much money you have to do it.

For an in-depth comparison of the most frequently used app development options—**Native, React Native, Flutter** and **Progressive Web Applications (PWAs)**—we've put together an overview that outlines what to consider when choosing the path best fit to your endeavor.

The following pages compare basic and advanced features, performance and speed, long term outcomes, cost efficiency and each option's ecosystem. We throw in some of our top engineers' tips as well, to share personal experiences and examples as an additional guiding light. (Based on their input, we've chosen to exclude the hybrid mobile app frameworks in this section, as STRV does not recommend them.)

Native

Native is the most familiar option, as it stems back to the beginnings of mobile development. It includes two platforms with their respective programming languages: Swift (or Objective-C) for iOS and Kotlin (or Java) for Android. Most businesses invest in native mobile app development because of the many time-tested benefits it offers—which tend to outweigh those of other options.

Pros:

- **Lightning-fast & responsive apps:** this means the best performance that you can achieve
- **Smooth user experience:** the app is created for a specific operating system
- **Top-notch UI:** the user interface and app flows follow each platform's UI standards
- **High user-to-customer conversion:** thanks to the points above
- **Leverages all platform & device features:** can use the latest features from day one
- **Close to the metal:** can easily access and use a device's hardware (GPS, camera, microphone, sensors, etc.)
- **Security:** most reliable data protection
- **Easier maintenance:** it's easier to manage two apps in two codebases than two apps in one codebase
- **Always up-to-date:** development tools are provided directly by Apple/Google

Cons:

- **Higher development & maintenance cost:** requires separate coding per platform
- **Operational & update-related hassle:** any fixed bug or update must be submitted to the app store and the app needs to be (automatically) updated

Why & When to Use

While the initial price tag is higher than with a hybrid approach, putting in those resources saves time and money in the long run. If building a customer base and strong customer loyalty is the core goal of your business, going native is the safest bet. Users love native apps. They're beautiful, blazing fast and they don't suffer from the uncanny valley of UI/UX that hybrid apps typically do. For instance, an iOS app should look, feel and behave like an iOS app. If it doesn't—as in the case of hybrid apps—users might not know what exactly is wrong, but they do notice.

Users love to use the latest and greatest features a platform offers. With native apps, you can immediately implement all functionalities made available in the latest operating system release (an advantage you often lose with hybrid apps), as well as the most up-to-date look and feel. Similarly, engineers appreciate that native development benefits from more than a decade of combined engineering experience and platform improvements. The amount of shared knowledge at your disposal is unparalleled. Whatever the issue, someone has probably faced it before—and has shared the solution.

If your business requires hardware integration, native apps are tightly connected to the underlying platform. Antennas, sensors, chips... it's all at your disposal. And all of it is immensely powerful. Should a new sensor be released on the latest devices, for example, you're covered.

> *Users love native apps. They're beautiful, blazing fast and they don't suffer from the uncanny valley of UI/UX that hybrid apps typically do.*
>
> *— Jan Kaltoun, iOS Platform Lead*

STRV Tip

STRV clients sometimes consider other options due to cost-saving. But ultimately, as development moves forward, engineers face obstacles that come at a cost. We've had multiple clients come back asking us to rewrite their multi-platform apps to native in order to provide a better user experience or to get access to device hardware. At this point, however, money has already been spent and the investment has been too high to just ditch the apps. A solution is to integrate native code into the apps, which means that those initial benefits of having one team and one language for both platforms are negated.

As for future solutions, some parts of native code can now be shared between platforms (Android and iOS). This works well with business logic or networking logic, which is usually the same on both platforms. Currently, there are two options: cross-platform Swift or Kotlin Multiplatform. This approach is becoming more and more popular and we believe it's the future of mobile development.

85%

85% of consumers prefer **native apps** to mobile websites. (Business2Community, 2018)

React Native

React Native is a cross-platform technology developed by Facebook that uses native components instead of web components as its foundation. It allows engineers to build a dual-operating system (Android and iOS) app using React and JavaScript, and aims to bridge the gap between web and mobile development.

Pros:

- **Time-efficient:** cross-platforms use one source code for both platforms
- **Outstanding developer experience:** declarative and predictable UI, hot reloading
- **Sharable codebase:** written in Node.js with web or backend
- **Lower cost:** one development team suffices, which also leads to easier project management

Cons:

- **Newer & less mature:** always undergoing changes and improvements
- **Slower & not as high-performing:** unable to utilize full capabilities of systems and devices
  Fewer resources & experts: the community is not as widespread

Why & When to Use

React Native is a good option for simple apps and prototypes, as it’s best if you need a quick, cost-efficient development process and aren’t toying with advanced features. Specifically, it works well for B2B apps, software used for internal processes within a company and apps mainly for filling out forms. Alternatively, it is a risky choice for apps that heavily use animations, leverage the latest operating system features or require high performance. Similarly, React Native is not a good fit for game development.

March 2015, **F8 Conference** –
Official launch of React Native

STRV Tip

The above-mentioned pros can be seen as temporary benefits only. Yes, cross-platforms use one source code for both platforms, but only until you need something more complicated—at which point you'll need native engineers. And while the cost is initially lower, it's more than likely that you'll eventually need to support all three platforms—JS, iOS and Android. These are the reasons why we typically recommend going native.

Flutter

Flutter is a free, open-source cross-platform framework developed by Google. Like React Native, Flutter requires only one codebase for both platforms (Android and iOS).

Pros:

- **Time-efficient:** cross-platforms use one source code for both platforms
- **Lower cost:** one development team suffices, which also leads to easier project management
- **Fast development:** changes in the code can be seen right away in the app

Cons:

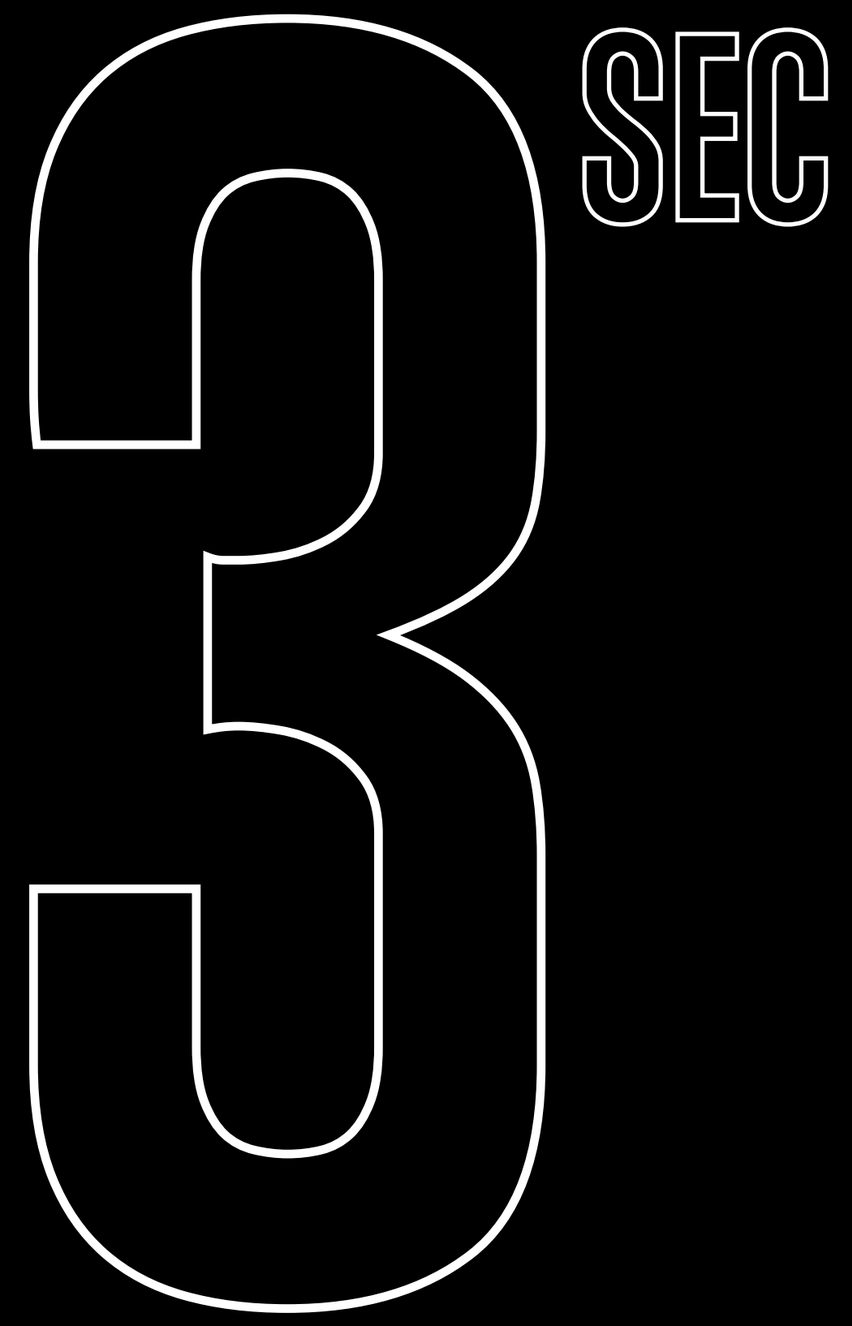
- **Does not use native components:** app design usually doesn't follow a platform's UI standards (utilizes primarily Google's Material design)
- **Newer & less mature:** always undergoing changes
- **Limited community & resources:** limited number of available third-party libraries and community support
- **Written in Dart:** Dart is not as popular and widespread as other programming languages
- **UI not updated when iOS/Android changes:** does not automatically benefit from OS improvements

Why & When to Use

If you’re looking to prepare an MVP for investors as soon as possible, Flutter is a solid choice. As a new technology with nice animations and smooth UI, it’s great for simple apps and building prototypes in record time—in part thanks to good Firebase support, which also gives access to backend services for mobile applications like authentication, storage, database and hosting, all without having to maintain your own servers.

STRV Tip

As mentioned, Flutter is a new technology, which means there are aspects that are yet to be explored. Add to this the risk of Google cutting it and there’s a sense of instability that makes us wary of using it on large, complex projects. There is clear potential, but we don’t feel that the benefits are worth the current risks.



The average Flutter app reloading time is **3 seconds.**

Progressive Web Apps (PWAs)

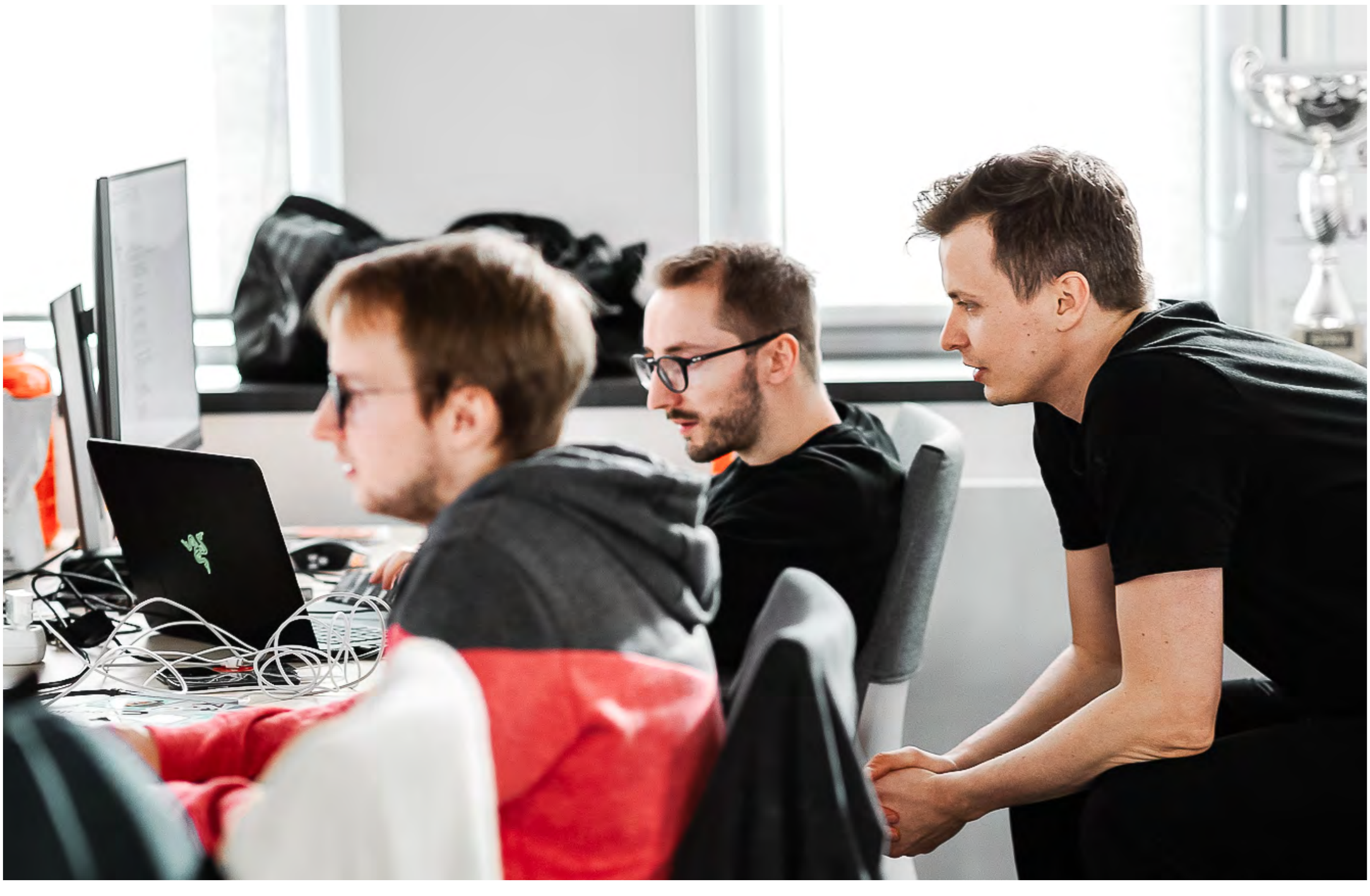
Unlike mobile apps, a PWA is an application delivered through the web and is developed using common web technologies, like HTML, CSS and JavaScript. It works on any platform that utilizes a standards-compliant browser. Simply put, it's an app-like web experience. And it's constantly improving.

Pros:

- **Lower cost:** same content as web and no app store updates needed
- **Easier development & maintenance:** one team of web engineers, one code
- **Offline browsing & quick loading:** retain most of their functionality when offline
- **Market reach:** easily accessible even with a poor Internet connection and reaches both iOS and Android users

Cons:

- **Limited hardware integration:** but gradually improving; camera, GPS, accelerometer or gyroscope are accessible
- **Limited integration with other apps:** WebShare API is capable of the same system-provided share capabilities as native apps, but cannot access, for example, the contact list
- **Limited possibilities:** fewer features; for example, iOS doesn't support all PWA functionalities (like push notifications)
- **Not native:** suboptimal UI and UX compared to native



Why & When to Use

For businesses that focus on text, images or video content as their primary offering (such as e-commerce, news, blogs and influencers), PWAs provide great discoverability, performance and SEO. These apps are also a good option if some of your users live in countries with a poor Internet connection, or if you are a startup; many startups want an iOS app first (for financial or other reasons), causing them to lose out on the Android market. With a PWA, you can reach everyone, regardless of which device they use.

STRV Tip

Something to note here is app distribution. Once your app hits the App Store, it is required to undergo the App Store's review process—which typically takes one day. This applies to native apps, react native apps, flutter apps, etc., but does not apply to PWAs, as they're basically websites with an icon on your phone's home screen. This can be seen as a vice because users cannot find your app in the App Store (although PWAs can now be offered through Google Play Store), which makes reaching a wider audience difficult. However, if your business touches on a topic or service that the App Store limits (such as vaping), PWAs are a great choice.

There are, of course, other options of distribution—such as Enterprise distribution on iOS or APK distribution on Android. And there is also another development approach worth mentioning here: WebAssembly, which is currently on the rise. By using native languages, WebAssembly allows web applications to run at near-native speed, letting engineers develop games, VR/AR or music apps. It's good to consider all options, weigh your priorities and only then make a choice.

15%

Mobile sites that load in
2 seconds or less have
a **15% higher conversion rate**
than the average mobile site.

BACKEND

Michal Klacko
Backend Platform Lead

Honza Hovorka
Backend Platform Lead

Introduction

Backend is a bit like the brain of the entire operation. A robust backend is necessary not just during the initial release, but for future iterations of the product. It is responsible for all of the data, data manipulation, the product's business logic, security and all heavy computation/data science, and it serves as middleware for applications (web and mobile apps). Get backend right and you're ready for any changes that your product may—and almost always will—require in the future.

From the very first product iteration, backend engineers should be involved. They play an important role in choosing the technologies and approaches that best accommodate every specific need. This means writing code that's designed to handle future features and extensions, even when they aren't yet known or defined. To do this, engineers need to comprehend a business top to bottom, an understanding that is then reflected in the code, database architecture and every decision made.

And it all starts with choosing the most well-suited **programming language, frameworks, database solutions and deployment and hosting.** Let's get into which options are best suited for which cases.

Programming Languages & Frameworks

There is no such thing as the best programming language. Each language is the right fit for a specific domain. Based on factors including popularity, maturity, use cases and development experience, we've decided to cover **Javascript/Node.js, Python, C#/.NET, Golang, PHP** and **Java**.

> *Get backend right and you're ready for any changes that your product may—and almost always will—require in the future.*
>
> *— Michal Klacko, Backend Platform Lead*

Javascript/Node.js (Typescript)

Due to the asynchronous nature of Node.js and the huge support of the JavaScript community, this technology is fit for common APIs and data aggregators. It is cost-effective and highly scalable, with significantly faster development time (compared to strongly-typed languages like C# or Java) thanks to an open-source online depository of more than 230,000 available modules.

JavaScript is a dynamically-typed language. For those who prefer strongly-typed languages, Typescript is a great alternative. Keep in mind that in terms of being a strongly-typed language, Typescript may not be quite on par with Java or C#, but it does have some of the advantages while still maintaining the flexibility associated with the JavaScript world.

code; instead, it is broken down into smaller modules. And last but not least, because Node.js uses the JavaScript programming language, you can share parts of the codebase between web and backend. This not only speeds up development, but comes with the benefit of having uniform technology throughout the platform.

Why & When to Use

Widely used by both growing and established businesses, Node.js is great for prototyping your business idea. It is perfectly suited for building real-time apps, chat apps, gaming apps and blog sites, and it takes a modern approach to the use cases which used to be solved using PHP. Additionally, it’s great for microservices that require a scalable solution because the logic is not developed as a large

37%

37% of all Node.js users are **backend oriented.**

STRV Tip

In the case of **simple backend**—such as API serving data from database to web or mobile apps or basic integration between various third-party services (e.g. payments, file storages, messaging, notifications, etc.)—we recommend using Node.js, as it's a very lightweight and flexible technology built for these use cases. And in circumstances where it's fitting or necessary to share code between a client and a backend team, Node.js is a great fit. At STRV, we use it for most of our projects.

Python

Python is a highly popular programming language that emphasizes code readability. This makes it closer to spoken English than its alternatives, allowing engineers to work quickly and to integrate systems efficiently. This is why, over the years, Python has become the #1 language when it comes to data science and machine learning—for which it currently has the biggest toolset. It also continues to be widely used for building webs, APIs and various standalone scripts.

Why & When to Use

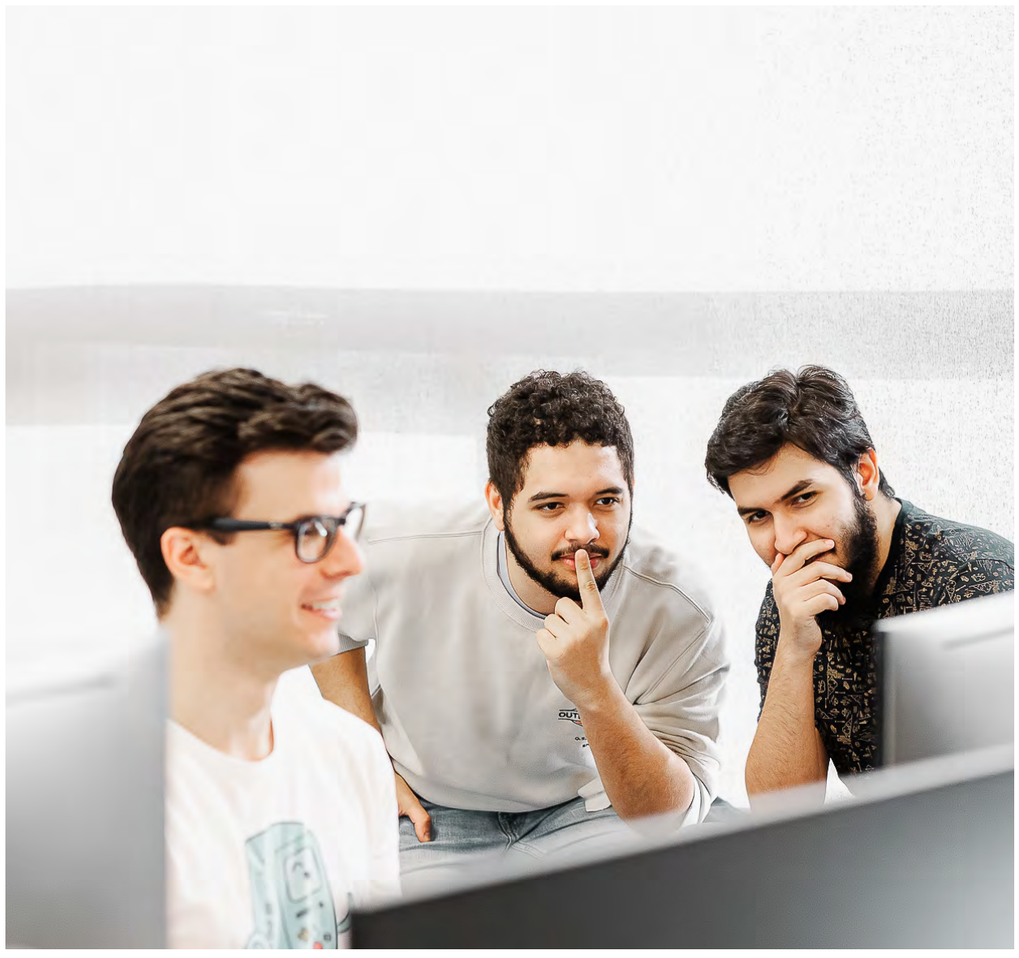
For businesses that employ both experienced and newer engineers, Python is a great choice because it is very simple and easy to understand. Its rich toolset is especially useful for machine learning, visualization and data analysis. It is also good for creating an interactive user interface for a business website or app (although, from our experience, still not as good as Node.js) and for helping with automating tasks. Thanks to its versatility, it’s useful for many other cases, like creating games, testing microchip viability and writing small standalone scripts for various data manipulations.

Python's rich toolset is especially useful for machine learning, visualization and data analysis.

— Honza Hovorka, Backend Platform Lead

C#/.NET

Another versatile programming language with a mature ecosystem, C# was developed by Microsoft and is used to perform a wide range of objectives across multiple industries. Because it is strongly-typed, errors are detected before an app goes live, which results in a much more stable solution. This is a huge benefit compared to dynamically-typed languages (such as JavaScript or Python), but it comes with its downsides—the worst of which is the significantly longer development time and less flexibility.

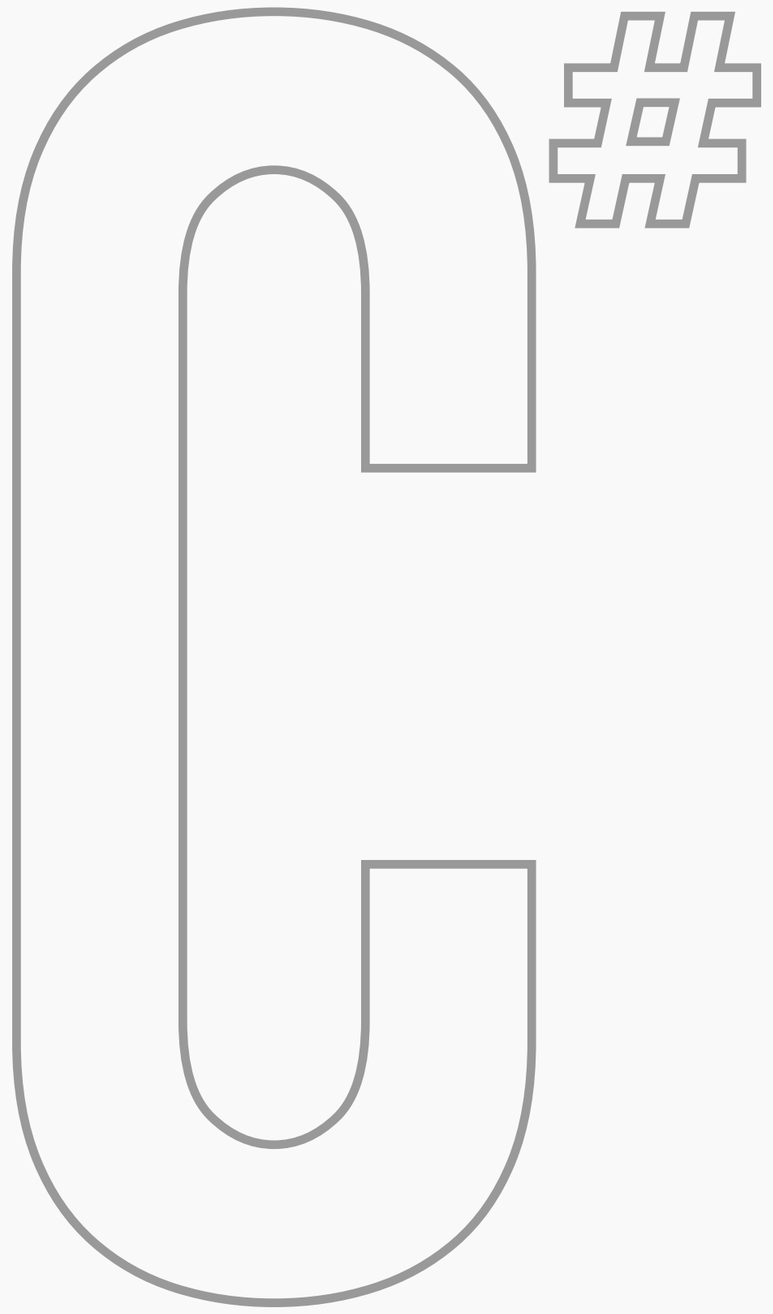


Why & When to Use

Applications that require more control and a stricter environment—such as enterprise and fin-tech applications—are always a good candidate for using C# due to the nature of its design. Additionally, C#/.NET now works with every other platform as well as it does with Microsoft. With C#, you can comfortably develop typical APIs for web or mobile apps. Generally, C# can be found in almost every field, including game development.

STRV Tip

Aside from the mentioned enterprise and fin-tech applications, we also recommend using C# when developing games in Unity.



The syntax of the C# language is similar to the C-style family —like Java, C or C++.

Golang

A programming language from Google, Golang is a newer, general-purpose language that is very easy to understand and was designed with system programming in mind. It is strongly-typed and has a growing ecosystem of tools, making it a good choice for larger collaborative projects. When creating Golang, Google released a Golang standard project layout (rules & policies) which most engineers continue to follow. This allows for huge versatility, as every engineer can easily understand your project.

> *Golang has recently gained a lot of popularity as a good choice when it comes to IoT, microservices architecture and performance-centric solutions.*
>
> *— Michal Klacko, Backend Platform Lead*

Why & When to Use

Golang has recently gained a lot of popularity as a good choice when it comes to IoT, micro-services architecture and performance-centric solutions. For IoT, it works well because it can be compiled into machine code, making it easy to run on almost any machine. Microservices benefit from Golang because it is lightweight and plays very well with gRPC (which is a lightweight format that's often used in internal communication between microservices). And its value for performance-centric solutions lies in how it compiles to machine code—which results in fast execution—and in how Golang itself allows low-level programming.

19%

19% of DevOps are planning to use Golang in the future.

STRV Tip

Because of how quickly Golang gained a following, it has begun to be used on solutions for which it isn't really the best fit—such as typical REST APIs. Compared to Node.js, Golang lacks certain packages needed to support all typical use cases for APIs. Nonetheless, we do occasionally utilize Golang or combine it with other languages, like when it comes to microservices.

PHP

The PHP language covers similar ground as does Node.js. It gained a lot of popularity due to its nature as an easy-to-use dynamic language and because when it came out, there wasn't anything similar available. PHP has undergone a lot of updates in the past years due to how web and mobile development has changed; it now supports writing APIs (instead of server-generated web pages) and basic scalar types.

Why & When to Use

As a server-side scripting language that offers engineers a multitude of instruments, easy insertion in HTML code and connection to MySQL and PgSQL databases, PHP is useful when creating dynamic web pages. Use cases include typical web applications like forums, landing pages and forms collection. Due to its popularity in the past, many engineers have mastered writing PHP code, which is a big part of why business owners choose to take the PHP route. But because PHP was created from the ground up as a tool for building web pages, it had to gradually accumulate functionality and certain language features. There's a lot that engineers must know about it to maintain a large codebase, making it difficult to scale.

STRV Tip

Compared to Node.js, PHP lacks both the ability to share code with frontend and the huge support of the JavaScript community (as well as the availability of third-party SDKs). Which is why we always prefer Node.js over PHP.

About 64% of **WordPress sites** use PHP 7.1 or lower.

Java

Java is an object-oriented, strongly-typed coding language. In many ways, it's quite similar to C#; however, Java was built as a platform-agnostic solution, while C# was initially tied to Microsoft technologies. Java code is compiled to bytecode, which runs on JVM (Java Virtual Machine)—allowing it to run almost anywhere. But because it's a virtual machine, it is slower compared to, for example, Golang running in native machine code. A huge benefit is the large standard library and support from Oracle, which makes it a very mature and stable technology for the enterprise and government sector.

Why & When to Use

As the most-used runtime platform on enterprise systems, it is commonly used for banking and fin-tech web apps due to its security, functionality and strongly-typed nature. In case you can't rely on third-party packages/libraries (for example, in the case of banking or fin-tech solutions), Java is a great choice thanks to its large standard library. And as it is backed up by Oracle, Java is very friendly with the rest of their technologies—such as Oracle Database. Additionally, Java is also used to develop Android apps (and is currently competing with Kotlin for this position).

Databases

Most developed apps need to persist data. That's where a database comes in. A database is an organized collection of data; it provides various functions to retrieve and manipulate data. There are multiple types of databases with various approaches. Two of the most-used types are **Relational (SQL)** and **Document (NoSQL) databases.**

Why & When to Use

These databases are the best option when it comes to structured data with strong relationships (which is most cases) and gathering statistics and correlations between data. The biggest disadvantage is that they are hard to scale with Big Data, or if you don't know the exact structure of your data. As long as this is not your case, SQL databases are a safe, strong choice.

Relational SQL Databases

SQL stands for Structured Query Language, which is the most commonly used database language. Although it is now 50 years old, it continues to be the most popular. "Relational" means that data is structured based on mutual relations; the main strength of this model is, at its core, the use of tables, which are an easy way to store and access data. It has long been the preferred type of databases. Relations together with SQL allow for the creation of very complex queries and for the retrieval of complex data structures and their correlations. A huge advantage of most relational databases is that the database itself takes care of data consistency.

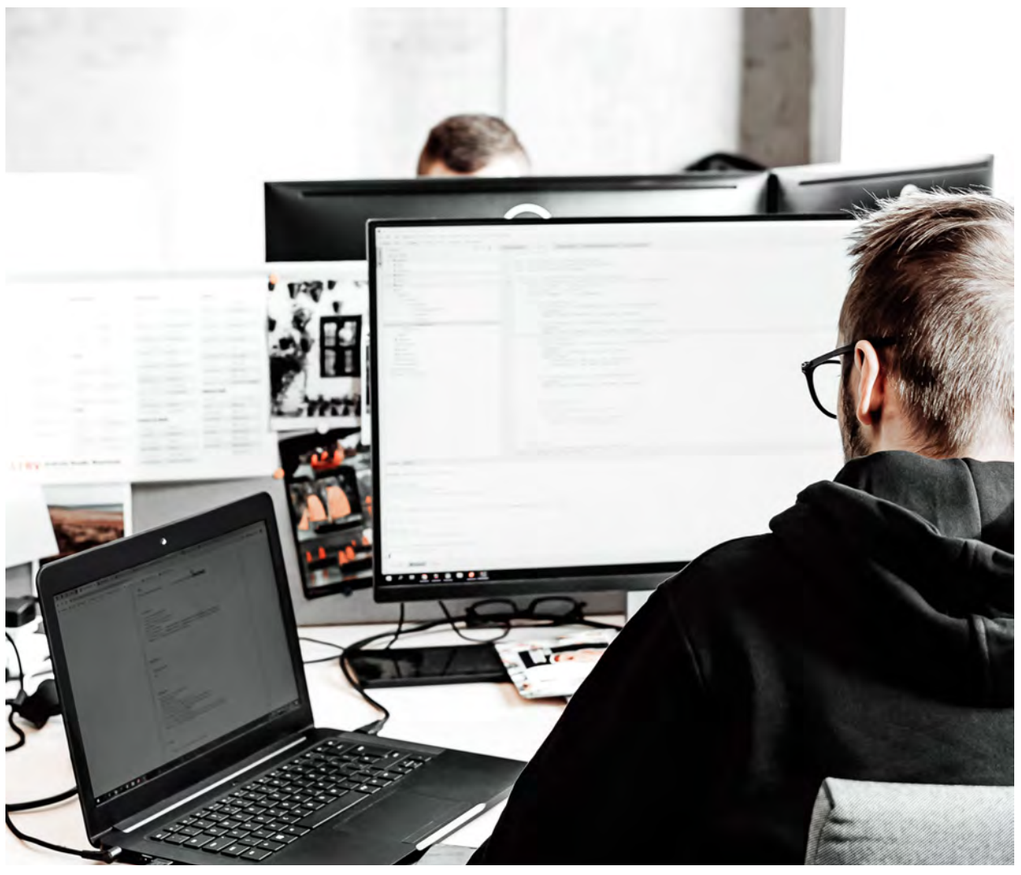
STRV Tip

Although we've worked with all major SQL databases—such as PostgreSQL, MS SQL, My SQL, Maria DB and Oracle—we prefer PostgreSQL, as it is the most mature platform-agnostic SQL database.

PostgreSQL has over 30 years of **active, reliable development.**

Document (NoSQL)

As the name suggests, document databases treat data as documents—meaning that every piece of data is a record stored within a collection. Because there is no schema, data can be dynamic. Document databases do not support relationships between documents; you can simulate them, but the databases will not ensure consistency. A big advantage of NoSQL databases is that because they are lightweight, they are very fast and versatile, favoring availability and speed over consistency.



Why & When to Use

When it comes to non-structured or highly dynamic data—like logs, analytics data, etc.—NoSQL is the way to go. There are multiple popular NoSQL databases, such as MongoDB, DynamoDB, Firebase, etc. Each of them has a different use case. For example, Firebase is for realtime data, DynamoDB for most simple key/value-based data and MongoDB for everything else (so to speak).

20M

DynamoDB can handle
20 million **requests per second.**

STRV Tip

When we need to work with Big Data, highly dynamic data or data with an unknown structure (such as logs or various third-party aggregations), we prefer NoSQL databases over SQL databases. Our favorite NoSQL databases are MongoDB, DynamoDB or, when working with C#, RavenDB.

Deployment & Hosting

The future of data and hosting is in the cloud. It no longer makes sense for businesses to deal with the hassle of having their own servers/data centers (unless they have strong reasons to do it, e.g. banks). Clouds also provide various services and tools that you can leverage for your business—from hosting, autoscaling and handling data (even big data), all the way to analytics, using machine learning models as a service and much more. Another alternative, usually for big enterprises, is the hybrid model: combining cloud providers with a company's own infrastructure.

For the deployment of backend applications, Docker is considered the industry standard of the modern era, as it creates a container for the application which contains everything needed for the app to run. Docker is supported by all major and modern cloud providers. You can also run it on your own servers; this is achieved by Docker's independence from other technologies.

Heroku

Heroku is a container-based cloud Platform as a Service (PaaS). Engineers use it to deploy, manage and scale modern apps. Heroku aims to enhance productivity by eliminating the distractions of maintaining servers, hardware and infrastructure. Its main strength is in its simplicity; just a few clicks and your server is running with hundreds of "plugins." Heroku natively supports a number of languages and frameworks: Node.js, Ruby, Python, Java, PHP, Golang, Scala and Clojure. It also allows you to run containerized applications using Docker; theoretically, it's possible with any language. While Heroku doesn't have its own data centers—most of its resources are created in AWS—this happens in the background and you do not interact with the AWS itself.

Why & When to Use

For small/medium applications, Heroku is the optimal "out of the box" solution. No DevOps (deployment and operations) and maintenance needed. All crucial Heroku resources run on AWS in the background, which means Heroku runs on one of the biggest and most battle-tested infrastructures in the world—yet it doesn't require you to deal with the complex AWS setup. This also makes a possible transition to AWS easier (once/if it's required).

STRV Tip

We strongly recommend to use this for prototypes, MVPs or simple applications, as it is lightweight and can be set up in literally just a few minutes. Heroku is relatively cheap in the beginning and for smaller applications, but it may become significantly more expensive in later stages of a product. It is important to note that with growing complexity, Heroku's simplicity may become a problem, as it does not allow dramatic customizations as well as other big players do (such as AWS, GCP, Azure).

Nonetheless, it's still possible to migrate later on. And what we consider to be the biggest advantage of using Heroku is that it doesn't require having a DevOps (development and operations) engineer to take care of it.

Heroku manages more than **2 million data stores.**

AWS (Amazon Web Services)

The biggest of all cloud providers, AWS provides IaaS (Infrastructure as a Service) and PaaS (Platform as a Service) for cloud ecosystems. This combination helps create a scalable cloud application without dealing with infrastructure provisioning and management issues. And because AWS is amongst the most popular cloud providers in the engineering community, finding engineers becomes that much easier.

Why & When to Use

Although it is the biggest and oldest of all clouds, AWS can be a bit time-consuming to set up compared to other options. Nonetheless, it offers the widest customization options and flexibility, and continues to be the most universal of all cloud providers in terms of the services it allows—all of which makes it a great solution for most use cases.

> *Because AWS is amongst the most popular cloud providers in the engineering community, finding engineers becomes that much easier.*

— Honza Hovorka, Backend Platform Lead

AWS spans 70 availability zones within **22 geographic regions** around the world.

STRV Tip

We use AWS almost exclusively when we feel Heroku might not be enough to handle complex architecture. Additionally, we use quite a few services (Analytics, ML models, etc.) provided by AWS. To avoid dealing with AWC Console—which is not a pleasure to use for engineers—consider using Terraform to manage your infrastructure/services.

GCP (Google Cloud Platform)

A cloud ecosystem by Google, GCP provides infrastructure and and Platform as a Service (PaaS), serverless computing environments, management tools and a series of modular cloud services including computing, data storage, data analytics and machine learning. The tools and services offered are similar to those of Amazon's AWS. It is worth noting that with GCP, you are indirectly leveraging advanced technologies and Google's know-how.

Why & When to Use

GCP is a great choice when working with either real-time data (Firestore), machine learning or Big Data, as those are the fields in which Google has long been dominating the digital world. If your business relies heavily on Google services such as YouTube or Google Search, GCP is the best choice thanks to its offer of native integrations. With GCP, you also get some of Google's technologies as a service—like image recognition or voice assistant.

STRV Tip

We use GCP—specifically Firestore—almost exclusively when we have to build real-time applications like chats. Many engineers prefer GCP because its UI/Console is more polished and well-arranged compared to other options.

85%

About 85% of IT managers use **public cloud infrastructure.**

Azure

Azure is an ever-growing cloud computing service created by Microsoft for building, testing, deploying and managing applications and services through Microsoft-managed data centers. It was originally designed mainly for other Microsoft technologies but has recently switched to a technology-agnostic approach and is now easy to use with everything. Azure is especially popular among enterprise companies due to relationships with and the influence of Microsoft.

Why & When to Use

While Azure offers very similar tools to that of other clouds, it’s the go-to choice when working with C# or other Microsoft technologies and if your company is using Microsoft tools internally (due to the integrated Microsoft ecosystem).

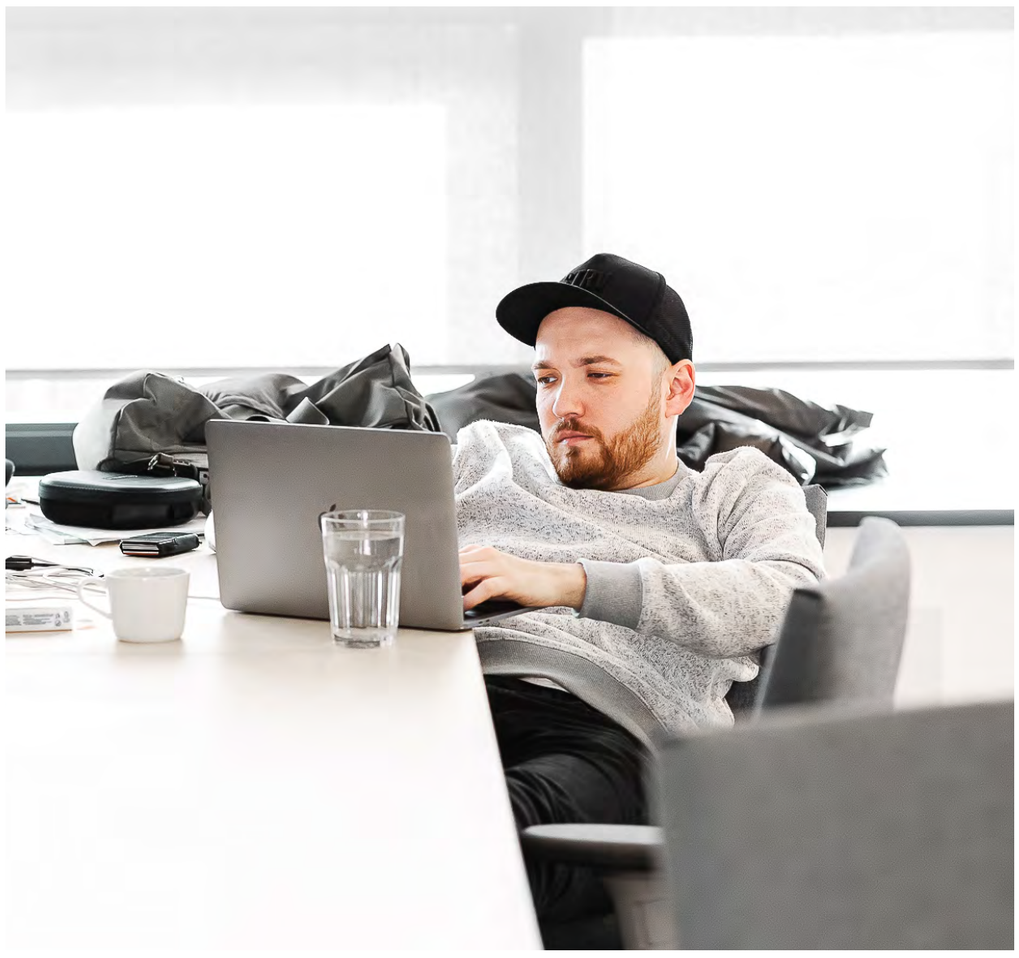
FRONTEND

Daniel Kraus
Frontend Platform Lead

Danny Kijkov
Frontend Platform Lead

Introduction

Frontend is the visual aspect of the web with which users interact. For this reason, it is seen as the most accessible way of presenting a company's product or service. It works on any device, operating system or internet connection but, compared to the native world, it has many limitations. Due to frontend's importance, however, web APIs are constantly evolving, getting stronger and providing more capabilities.



Why & When to Use

The main benefit of web applications is discoverability. You can easily search and share them. All you need to know is the URL of a website; you don't need to buy or install it on your device. This simplicity makes websites fundamental to all businesses, which is why web technologies are usually a part of any software solution.

The basics are a landing page, CMS, administration and/or emails. But there are endless ways to bring websites to life. With the rise of single-page app frameworks, for example, it's now possible to build enterprise-grade apps. And progressive web apps allow building offline experiences for websites, enhancing its functionality with push notifications and much more.

Due to frontend's importance, web APIs are constantly evolving, getting stronger and providing more capabilities.

— Danny Kijkov, Frontend Platform Lead

STRV Tip

Over the past decade, we've been chasing ourselves to distinguish the best framework. Due to the progress of the whole web community, there are numerous great alternatives you can choose from. Our conclusion is, it doesn't matter what you choose. Or, rather, that shouldn't be your focus. What matters is: What will be the output? So, instead of delving into technologies, we're going to discuss the output.

94% of users form an opinion of a company's **credibility based** on its web's **visual appeal.**

Five Focus Points

The quality output of web apps could be defined by five categories, all of which work together to create a truly valuable result. These categories are:

- **Great UX/UI**
- **Stability**
- **Performance**
- **Accessibility**
- **Security**

Getting all of this right results in more than meets the eye. It’s crucial to understand the business impact of these focus points and why an investment in making them better means an investment in long-term success.

Great UX/UI

First impressions matter. Perfectly tuned UX/UI helps increase conversions, and automated end-to-end testing with visual regression can guarantee that you are not breaking your UI with new changes.

Stability

Having a crashing app can be detrimental to businesses. Using automated tests and monitoring services to pinpoint any problems before they got into production and cause real damage is always a must.

Performance

Get ahead of your competition by having a web app that loads faster. There are many studies covering how page load time affects conversion rates. Around 2014, mobile usage took over desktop use; since then, we've needed to focus on mobile first. And mobiles are unfortunately prone to be on a slow network.

Accessibility

The web can and should be accessible for anyone, despite any disabilities. Unfortunately, this optimization is not very common, yet it can make a big difference to your business. It’s simple: the more users feel catered to on your website, the more will become loyal customers who spread a good word.

Security

A project's security used to be considered backend's responsibility. Today, we know that the number of vulnerabilities present on frontend's side is just as high—especially with single-page apps, where you often deal with sensitive data. The bare minimum is to run on a secure https protocol, but a strong frontend engineer will find many more ways to ensure security.

Measuring Tools

For an even deeper understanding of what the above-mentioned focus points mean, it's very important to measure them. Because if there is ever an issue, you simply can't fix what you don't measure. We recommend utilizing the following tools:

- **Lighthouse:** great built-in tool in Chrome focusing on performance, accessibility, best practices and security
- **Chrome User Experience Report:** provides user experience metrics for how real-world Chrome users experience popular destinations on the web
- **WebPageTest:** runs a speed test from multiple locations from real browsers
- **Analytics:** always have them in place to better understand the behavior of your users
- **WAVE** and/or **Accessibility Insights for Web:** use to evaluate accessibility improvement opportunities
- **Web Vitals:** provides unified guidance for quality signals, helping to focus on the metrics that are essential for a healthy site

As for guidelines, that is a very broad topic. A solid foundation and strong development codex are the alpha-omega of quality software development. Squeeze the most out of your project setup. Choose the project starter wisely, consider using static type checking, set up code quality tools, enforce good practices, write tests, agree on code versioning flow, set up continuous integration & delivery and consider doing isolated component development.

For a more in-depth understanding, please see strv.gitbook.io/frontend/project-guidelines.

60%

In 2018, the **most regularly used frameworks** were React at 60%, Vue at 33% and AngularJS and Angular at 21% and 20%.

STRV Tip

At STRV, we are fans of React for numerous reasons. Nonetheless, we can easily adapt to Vue, Angular, Svelte or whatever comes next. As mentioned, we believe it’s really not about the road you take, but the output you end up with.

QA

Irina Shirnina

QA Platform Lead

Introduction

QA (Quality Assurance) testers are an extra pair of eyes for engineers, designers and product/project managers. Their main function is to prevent and catch defects before they become major, costly issues, thereby ensuring the quality of the entire development process and its result. This comes down to identifying all potential and present bugs, inconsistencies and other threats that could affect a product's or project's quality. Repairing every last flaw before releasing a product to the public is crucial because even the slightest glitch can impact business goals—according to a survey, 62% of users uninstall an app immediately if they experience a bug.

While top designers and engineers fervently check their own work, they are still dealing with deadlines, expectations and responsibilities that can affect how many minor details are missed. Especially when there is a need to explore the product in-depth, rather than going over and confirming the aspects that have already been clearly outlined.

The Value of QA

It always costs less to prevent a mistake than to fix it. This is why, ideally, testers are there for the whole ride. They're validating whether users want or need the product, helping define project requirements, providing valuable and timely information about the actual state of a product and ensuring that it is user-friendly and functions as it should. And because they've worked with apps across many industries, a skilled QA team often goes above and beyond by bringing new ideas on how to improve the entire product.

The value of QA is preventing the financial, time and business costs of mistakes. Bugs are an inevitable part of development; no matter how brilliant the engineers, they are still human beings who make little mistakes. For business owners, it's not about hoping that everything goes right. It's about realizing some things will go wrong, and dealing with this reality in the most efficient way possible.

- **Bugs cost money and time spent fixing them.** The faster that bugs are found, the easier and faster it is for engineers to fix them. This helps to minimize project delays and cost.
- **The daily rate of testers is less than that of engineers** in general, so it is more cost-effective to use a tester than to spend an engineer's time on testing.

- **Engineers can never test their own products as well as testers.** Proper testing needs a different way of thinking. Psychologically, engineers tend to treat the product kindly and follow the proper user flows while testing whether or not everything works. Testers do the opposite, so they usually find hidden issues faster. And having a fresh set of eyes is always useful.

Why & When to Use QA

The earlier testers join the process, the more they understand it and the better QA service they provide. If you're building a new product, it's important to have as many answers and statistics as possible. Utilizing QA from a project's early stages provides greater certainty in terms of every decision made. While the scope and details of a project are being confirmed, testers are already gathering all relevant information and creating the foundation of a Test Strategy document. This document evolves and needs to be modified throughout the project's lifecycle, always highlighting the latest and most up-to-date information about any testing done on the project.

STRV Tip

The best, most reliable QA team is one that is passionate not just about the work itself, but about your product. Testers should be assigned to a project based on their areas of interest, as well as experience. Put them on a project that they enjoy on a personal level and their input becomes a crucial part of the app's success.

Additionally, the testing process should be highly organized. At STRV, we have a five-stage process that starts at the commencement of the project in either the Design or the Development phase. What kind of tests are performed is outlined for each stage, but the final choice of tests depends on each project's scope, needs, timing and budget. It's the responsibility of our testers to define and explain the type of tests needed during a test strategy session before any testing begins.

Another aspect that requires defining is the devices and browsers used for testing. For example, when our team worked on DashNexus—a Dash cryptocurrency community platform based on blockchain—there were some rare combinations of devices and browsers used for testing in addition to the default set of devices. This was due to a highly specific target audience; some users in this community use Firefox on Android (which is quite uncommon).

30

Fixing bugs in later stages of a project can be 30x **more expensive than fixing them** during the prototype stage.

PRODUCT & PROJECT MANAGEMENT

Evan Sendra
Head of Delivery

Introduction

Product and project managers ensure that the team of designers, engineers and/or QA testers on a project are shipping the best possible product within project constraints. Product managers define the product, concerning themselves with what exactly will be built and why. Project managers drive and oversee the actual execution, taking care that work is on time, on budget and everything is properly tracked and communicated to the client without delay throughout the entire engagement.

The Value of Product Managers

During the early stages of a project, it is the product manager who takes on the responsibility of defining the product, its primary purpose and its target audience. A revolutionary idea is a great starting point, but there's a lot to figure out before moving to design and development. It is a product manager's job to delve into every last detail with you, asking questions that tease out assumptions and establish the problem that's being solved. Using research, user testing, experience and professional insight, a product manager turns an intangible idea into a plan with realistic expectations, which allows for an informed cost and time estimate.

Being able to predict the needs, outcomes and future iterations of your product dramatically improves your project's probability of shipping something valuable. Without prior experience with building a product from the ground up, this is a very intricate, often stressful task to take on. A product manager guides you to the right questions, fully prepares you for development and ensures time won't be wasted during execution.

Why & When to Use Product Managers

For a product to hit the market strong, every detail must add value to the overall purpose. No rushed decisions, no guesses just to "get it done." Design must be tailored to the target audience, the development process leading up to the release must be impeccably planned and the entire production team must have a common vision. These are points that stem from the very first stage of a product's inception—the strategy (or discovery) phase. That's where you'll find product managers, and it's best to include them as soon as possible so that decisions don't start being made before you've gathered the necessary information.

The Value of Project Managers

Project management encompasses executing every aspect of a project, dealing with obstacles and leading the team and the work they do to a successful finish. Together with you and, if applicable, with a product manager, they build a roadmap that considers all possible speed bumps, is malleable enough to not limit creativity and guarantees everyone is aligned in working towards the vision, not to mention the timeline and budget.

Apart from actively organizing and taking charge of tasks and work iterations, a project manager must also deal with people's ideas and emotions. Various creatives may clash against each other, and it is very important to dedicate time to resolving these situations and handling them with care—all while ensuring that the budget, timeline, team productivity and final results are in no way affected negatively. Having a professional project manager in charge means having a trustworthy project leader that understands what's at stake and is a skilled collaborator. As a client, you feel confident that you have in no way given up your power to someone else; all decisions are a conversation and you are constantly updated about the project's progress, successes and setbacks.

The distinct experience a project manager has is invaluable. While expert designers and engineers are undoubtedly reliable in doing their work, things come up; for example, features may need to be added or removed, and it's important to see the big picture of how each change may affect the outcome. If you have a budget of $100k, a skilled PM will manage things in a way as to not jeopardize the plan. One in six IT projects have an overage of 200%, and the best way to avoid being a statistic is to have your project led by someone with a track record of avoiding pitfalls.

It is a product manager’s job to delve into every last detail, asking questions that tease out assumptions and establish the problem that’s being solved.

— Evan Sendra, Head of Delivery

Why & When to Use Project Managers

In particular, why and when is it beneficial to use a project manager from the same agency/company as your designers and/or engineers? This depends on the size of the team and the state of the project. The bigger the team and the project, the more it is advisable to utilize a project manager that is close to the team members. Having a project manager who comes in already understanding the process and needs of everyone on the team, knowing how they work together and anticipating their reactions saves endless time. Both sides feel comfortable and open, which knocks down all barriers and allows for exceptional cooperation—and, subsequently, an exceptional product.

There are many questions that come up as design and development move forward. A project manager is open to a product evolving but is always alert, securing that the project's pillars remain intact. Are things getting done efficiently? Are problems being tackled properly? Is the team functioning well? And if, for example, an engineer presents an obstacle, is it a performance issue or does it go deeper, stemming from a project flaw? All of this requires a project manager's full dedication during all stages of development.

Recommended Tools

When it comes to various software development tools, we've found Jira Next-Gen to be the best option for our team. Jira Classic may seem like the go-to because it's so powerful and has so many features, but it has proven to be pretty cumbersome and difficult to use not only for product and project managers, but also for engineers and clients. From a business ROI perspective, how Jira Next-Gen really hits the sweet spot is that it's the simplest to use for software teams.

STRV Tip

At STRV, our product/project managers are skilled in handling both roles, meaning they can transition from one set of responsibilities to the other smoothly. This also negates all delays that come with a handover of information between two people.

In terms of how product and project managers approach their work and organize work iterations, we generally recommend the agile methodology and two-week sprints. Within each sprint, we start with backlog grooming and planning, then proceed to have daily written updates from the design, engineering and QA teams, ending the sprint with a review of what is complete and, ideally, a demo of what has been accomplished. The key benefit of this methodology is that it's the right trade-off between the client receiving insight into the development process, and the engineers and designers having enough time to fully focus and produce something meaningful.

As a final note, we believe that Scrum—the popular agile process framework—is both a great tool and one that can hurt a business like ours. At STRV, we don't do Scrum by the book because it's highly specific, with a lot of rules that would limit our flexibility. Which is why we have our own take and approach based on Scrum, but fluid enough so that we can adapt when clients want to do things differently.

Agile project management as a concept has been around for over 20 years.

Final Note

No amount of insight guarantees success, but it certainly raises the probability of every decision being the best one possible. That's what we set out to do when we started this ebook. Our senior engineers, QA testers and project and product managers collaborated to summarize years of professional experience—and to produce a valuable, honest and comprehensive guide. We hope it helps you create something you'll be proud of for years to come.

And a final piece of advice from all of us at STRV: We wholeheartedly believe that having a reliable partner by your side is an invaluable asset. Take the time to get to know the team. Ask questions. Check references. Find people you can trust to do what is right for your project.

If you would like to find out if we are the partner for you, we'd be thrilled to talk.

Just reach out.

THIS EBOOK WAS LAST UPDATED ON JUNE 1ST, 2020.