May 29, 2025

Why We Keep Choosing Flutter

TECH SPOTLIGHT

Written by: Max Hoaglund, Director of Technology

Clientek makes a conscious effort to avoid loyalty to specific languages or frameworks when building custom applications. We don't want to ever be in a position where our insistence on a given set of tools negatively impacts our ability to deliver what our customers need. We keep track of the high-level features we deliver and capture the pain points that emerge. This is closest we come to allowing our past experience to drive our choice of implementation framework. It's not exactly empirical but it's also not totally based on sentiment.

Counter-intuitively, this philosophy has led us down a road where we often end up using a consistent set of technologies. Clientek developers embrace a posture where they deeply understand one or two frameworks but also understand how to adopt a new one.

In the case of mobile app development, Flutter has emerged as the consistent framework of choice. I'd like to share some details about how and why we continue to use this framework, but this article is less about the distinct advantages of Flutter and more about how and why those advantages are often relevant to us now.

To use a tired simile, choosing a framework is like filling a jar with stones. Certain drivers of choice are big rocks, and others are gravel or sand: still significant but only considered once the big rocks are taken care of.

Big Rocks: Feature Alignment

As much as we can, we let the features we intend to develop influence our choice of platform. Many of the apps we build require deep integration with native

libraries and OS. We also frequently create apps with rich, interactive interfaces—think maps for navigation or complex data visualizations. While this general grouping of features are *supported* on most platforms, Flutter tends to make them easier to implement as opposed to React Native or Maui.

Flutter provides *platform channels* for calling native code directly from the Dart code that makes up the majority of the app, and this is simpler than React Native's JavaScript bridge. It's one less layer of abstraction. It also provides advantages for delivering rich functionality in a performant way—a Flutter app running on a client device operates entirely within the context of native code. React Native apps leverage a JavaScript VM (like V8 or Hermes) which interoperates with native code, and this introduces overhead. This overhead wouldn't be noticeable or concerning in an e-commerce or data entry app, but can make a big difference for a navigation app.

Gravel: Developer Experience

One of Clientek's goals is to offer extremely quick time-to-market for the applications we build for our customers. That means getting started fast, and any advantages we can get from our development platform are valuable to us. Flutter's use of a unified CLI for building and debugging is one such advantage—developers install it once, learn it quickly, and don't get pulled into a build tools rabbit-hole.

We've made great mobile apps in both Maui and React Native but have always spent more time nailing down the build process. Using those frameworks, our teams often need to learn more about the underlying tools and manage dependencies for them by hand.

Sand: Language Complexity

One thing that has caught my attention from overseeing Flutter development is that the volume of code needed to implement a given feature is higher than it would be in Maui or React Native. Dart and its Skia-based widget tree are detailed and verbose. This might sound like an interesting observation, but it isn't! Volume and rate of code change aren't really correlated with healthy or successful project delivery, and we've seen this proven out over decades of software development.

The level of challenge or legibility of a coding language isn't a nothing-burger. However, it's a dimension of custom

app delivery where people often quickly develop visceral reactions and assign too much importance to them. We don't have a lot of interest in mechanically or aesthetically elegant code. We also don't see a lot of significance in the principles that drive the design of coding languages. We don't ignore either of those things, but we make a point of prioritizing what our customers need and choosing the tools that align with that.

At the end of the day, our choice of Flutter is rarely about a single standout feature. It's about how the combination of Flutter's strengths aligns with the challenges of the projects we take on. The point isn't that Flutter is always the right choice—it's that we stay open-minded, weigh the tradeoffs, and let the needs of the customer guide our decisions.

CONTACT US