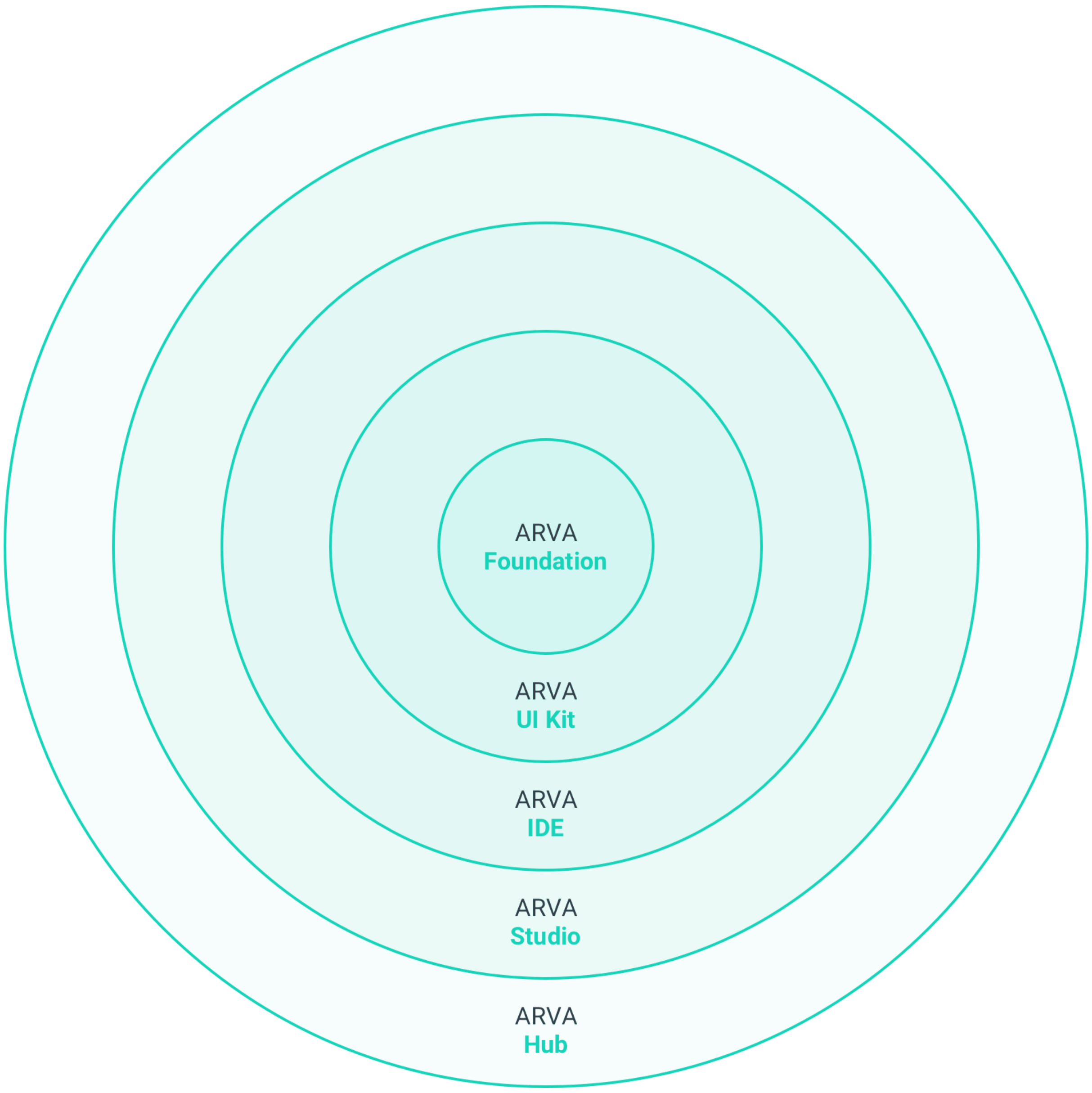
ARVA Architecture Overview

The ARVA ecosystem consists of a number of parts, successively building on top of each other. We will elaborate on the open-source technology chosen to develop specific software solutions.



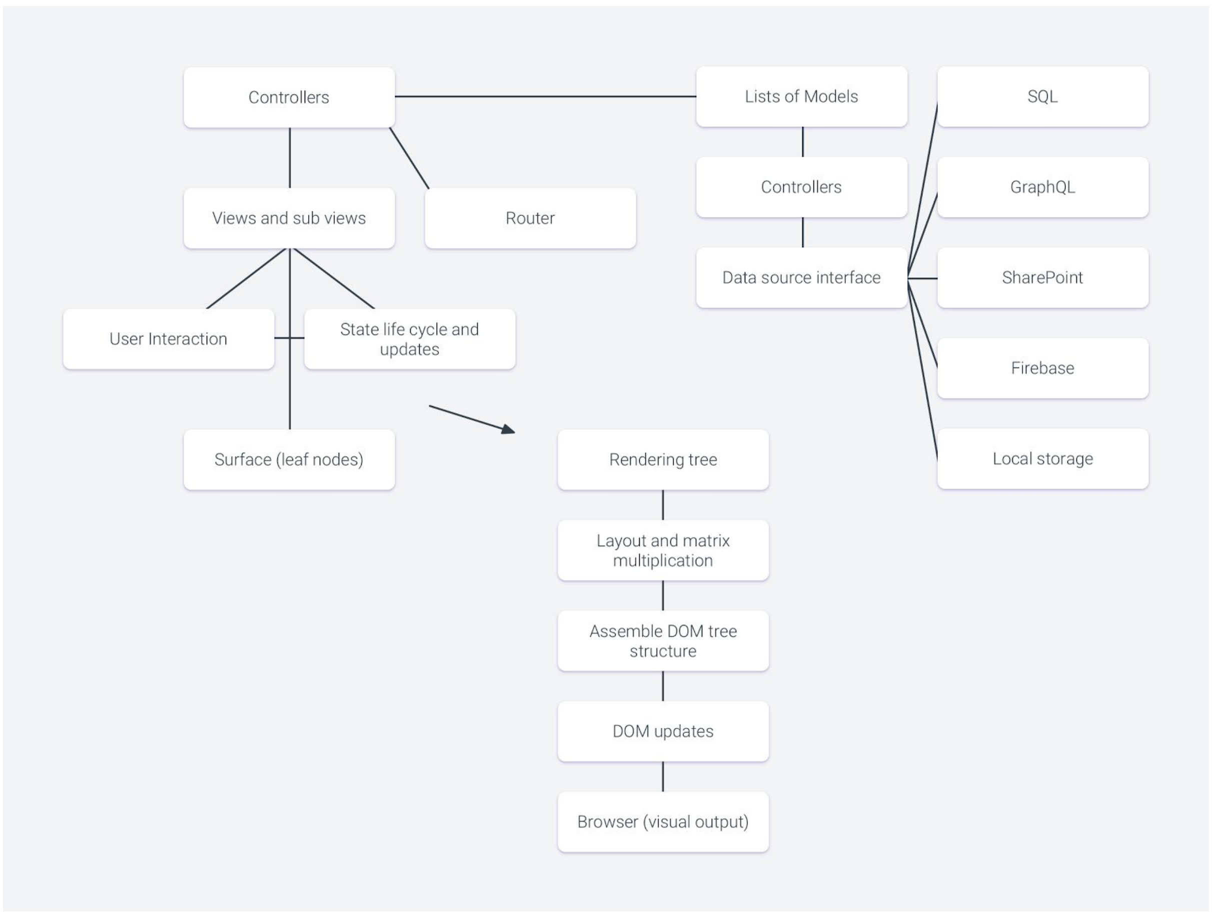
ARVA Foundation

ARVA Foundation uses a fork of a rendering engine called Famous, improving upon Famous' fast algorithms that render animations in a novel way. This abstracts away the commonly abused Turing-Incomplete languages such as HTML and CSS for the developer to further simplify the development process. To read more about the way ARVA has vastly improved and redesigned the initial effort of Famous, you can visit here.

This framework is also going in the opposite direction of many other frameworks, in terms of embracing standards. A common mistake mainstream actors within this sector make is to define their own ad-hoc definition language for the problem at hand (JSX, Handlebars, etc), which necessitates bridges to transpile into standard languages. We chose not to use these and instead keep up to date with the pace of ECMAScript standard committee.

In addition to going off the beaten path, we also follow. We adopt best practices and design patterns that have been proven to work well through decades like MVC and combine it with reactive programming, being a hot newcomer to the world of design patterns.

Figure 10: ARVA Foundation architecture



ARVA UI Kit

The UI Kit is a collection of ready-made components in Foundation. Since the technology stack is entirely based on Foundation, there’s not much more to say about the architecture.

ARVA IDE

The ARVA IDE will essentially consist of two parts: The editor itself, and the specification language to enable rapid development. To this end, we springboard from two existing open source initiatives:

- **Atom (https://github.com/atom/atom)**

Atom will be the heart of the coding environment (Code-Behind) and can be forked and altered without any restrictions because of it’s MIT license. The plugin architecture helps us develop the ARVA Studio features to bring creatives closer to the world of coding.

- **Autocode (https://github.com/ctate/autocode)**

Inspired by spec driven code generation we have forked the autocode project from Chris Tate to extend in such a way we can create a spec language for UI and Animation.

ARVA Studio

ARVA Studio relies on all preceding parts of the ARVA platform. It uses ARVA Foundation at its core. ARVA IDE is used for its project specification language and matching cross-platform code export. We will also have a special bridge to define customized studio-specific settings within the source code of the component. This will be used for utility within many of the UI kit components. We internally utilize the same specification language that was used in the ARVA IDE, for code generation.

ARVA UI Kit greatly helps project design and ARVA Hub is integrated to provide virtually limitless amounts of extra content.

ARVA Hub

The underlying version control technology ensures that ARVA Hub automatically creates branches for separate components and merges those branches into the project’s general branch without any merge conflicts or loss of work.

Aside from monetizing and consuming content from other digital creatives, ARVA Hub enables co-creation of products for teams. This co-creation is made possible by Git repository hosting services such as GitHub, Bitbucket and GitLab.