WHITE PAPER

Java application modernization:

Selecting the right approach

vaadin}>

Contents

Introduction

Over the last decade, the drive towards digital transformation and cloud adoption has been heavily focused on consumer-facing web applications as well as the use of SaaS packaged software. However, enterprises also rely on custom applications that are often the backbone for delivering the unique value of their particular business.

These business-critical applications, often built in Java, enable organizations to deliver differentiated products and services to their customers. Examples abound in every industry, including software that manages manufacturing processes, financial transactions, health care delivery, data analytics, and client support.

As organizations continue their digital transformation journey, they must consider when and how to modernize these Java enterprise applications. Through modernization, a company's software applications can better work with new web technologies, cloud services, mobile devices, and security processes, helping to eliminate the expensive technical debt that creates a drag on productivity. At the same time, modernizing the user experience (UX) for a legacy application can significantly improve employee productivity, especially for software that is built for repetitive activities across many users.

In this white paper, we will highlight the key motivations, considerations, challenges, and approaches to modernizing Java-based applications.

Software modernization is a top priority for 40% of IT sector decision makers according to a 2021 study by Forrester Consulting and IBM.

Motivations to modernize

There are different reasons why organizations might consider modernizing their applications.

One common reason that people consider modernization is technical debt. They have legacy or older applications that might be running on outdated technology that could be difficult to upgrade for new use cases or capabilities.

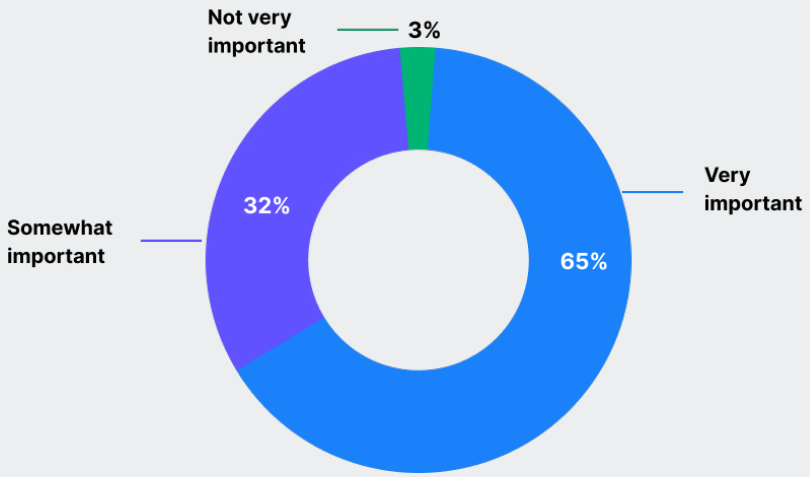
They might be on a digital transformation journey and need to shift the application out of a data center that's being decommissioned into the cloud. Or they may be outsourcing the hosting of the application to a third party. As a result, they need to move a desktop front end to a web front end and deploy that application to the cloud.

In addition, remote work is accelerating this transformation as it has sped up the need for digital transformation. For example, workers that moved from working in an office to working at home might need to access their enterprise application over the web and internet. This would require modernizing the application.

While the initial impetus for modernization might be technical debt, cloud migration, or remote work, there are three key business factors that a CIO would consider in deciding to proceed with modernization. These are risk, agility, and cost.

In a 2021 IDG survey of 400 IT leaders, 65% indicated that it was very important to accelerate application development and modernization to enable innovation, and advance the business.

Q: How important is the acceleration of application development and modernization as a means of enabling innovation and advancing your business?



Why application modernization is at the top of the CIO agenda

Business agility

Businesses are able to respond quickly to new market opportunities and innovate when their business-critical applications are flexible. Research shows that up to 84% of CEOs believe innovation is critical for growth.

However, legacy applications running on older technologies can resist change by being difficult to run on the latest platforms. Companies looking to reduce their dependence on older technologies are also reluctant to invest further in their legacy applications, even if they are business-critical applications.

Cost of ownership

Organizations can save money when modernizing their legacy applications: Embracing open-source software, reducing complexity by consolidating platforms, leveraging economies of scale in the cloud and maintaining a simpler, modern application architecture can all reduce the total cost of ownership.

Research shows legacy systems can climb 15% in ownership cost year to year, implying the budget-conscious decision would be to suffer the one-time expense of modernization in favor of creeping maintenance costs.

Risk

Security tops the list of technical risks that CIOs face in 2022. However, they are also concerned about business risks, such as downtime and business continuity. These can be associated with outdated or unsupported software.

Legacy technologies that no longer receive critical security updates can become increasingly vulnerable to cyber threats. Whether a technology comes from a vendor or an open source community, if it is deprecated, poorly maintained or no longer gets security patches, it can create significant business risks to the organization. These risks can be mitigated through application modernization and the use of technologies that are actively supported with regular updates.

Why UX matters: Driving efficiency for users and business

Good UX is about more than “being nice to users". Its main goal is to help users do what they need to do with an application with as little friction as possible. Reducing friction means that users can complete their tasks more quickly, increasing their efficiency.

Real world efficiency gains at Turo Italia

Turo Italia used Vaadin to build a new UX for a Quality Control application. Reporting issues that had in some previous cases taken them an hour to complete, now only take a few minutes. The substantial increase in efficiency is possible thanks to the user interface.

"Quriosity is an intuitive and easy application; it perfectly meets the needs of operators by making information quickly available. The user interface is so clear and well-arranged that it can be used also by people not so skilled in informatics.
As a result, we have considerably reduced the time necessary to create new non-compliancy reports and we can better keep control of our job."

Technology also plays a significant role in how employees engage at work. According to Forrester, employees who scored in the top 20% of their EX index – used to measure employee engagement – were more likely to be satisfied with their technology environment. Higher engagement leads to higher employee retention.

However, delivering a good UX isn't just about making users happy, it's also good for business. In fact, the McKinsey Design Index showed that companies with good design deliver revenue growth as much as 2x higher than others in their industry. Good UX also lets employees be more productive while reducing the risk of errors. For instance, Citibank's recent $500 million UI blunder is a classic example of how poor design can impact business negatively.

"Employees who are engaged are more likely to stay with their organization, reducing overall turnover and the costs associated with it. They feel a stronger bond to their organization's mission and purpose, making them more effective brand ambassadors. They build stronger relationships with customers, helping their company increase sales and profitability."

Forrester

Why DX matters: Driving efficiency for developers

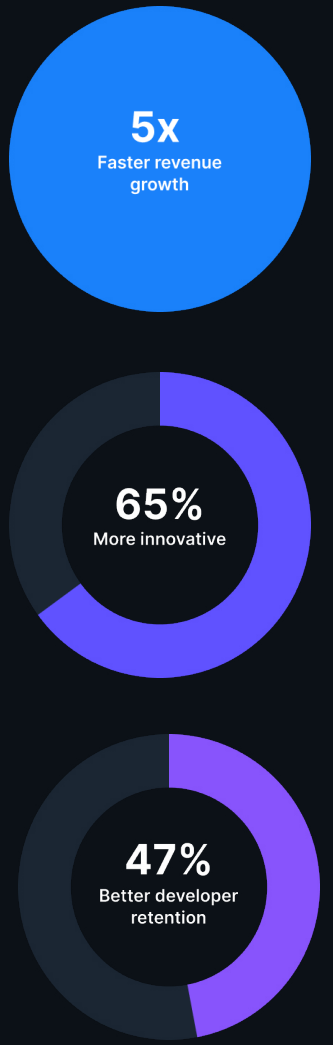
The benefits of application modernization are clear, yet CIOs face challenges in delivering these projects. A recent Mckinsey report reveals that 28 percent of respondents cited the complexity of their current environment as a key challenge. The report quotes a technology leader in financial services: “We were surprised by the hidden complexity, dependencies and hard-coding of legacy applications, and slow migration speed.”

Providing the right tools for developers will drive greater developer velocity. Higher developer velocity means faster time to market and, ultimately, higher revenue growth.

In fact, a McKinsey Developer Velocity report identified that those companies that have the highest Developer Velocity Index will see revenue growth up to five times higher than the rest of the market. In addition, they identified that companies with strong developer tools are 65 percent more innovative and have 47 percent better developer retention rates.

Many organizations will choose to combine their existing Java back-end services with a modern user interface. To ensure a timely transition and reduce future maintenance overhead, organizations need to carefully consider the tools, frameworks, and architectures they use to modernize their Java application

Impact of better developer tools and velocity



Choosing the right approach to modernize each application

There are two main technical approaches to modernizing Java applications – one is to use Java for the entire application and the other is to separate the front-end from the back-end. In the latter case, you will commonly choose a front-end language (such as JavaScript or TypeScript) and a front-end framework (such as Lit, React, Angular, Vue, etc) while continuing to use the Java-based backend. However, before you begin to decide on your technical approach, it's critical to understand the characteristics of the application, (including the key metrics that are critical to optimize for its success) and the characteristics of the development team.

Assessing application characteristics

Broadly, there are two major categories of applications to consider. The first would include "consumer-scale" applications that may serve up to millions of users. Examples include shopping, social, or consumer financial applications. In these applications, it is critical that the user interface be intuitive, since it may be days or weeks between user visits. It must also have a fast startup time, since consumers may abandon it if they are faced with initial waits.

The second category would include "enterprise-scale" applications that are often designed to serve hundreds to tens of thousands of users. These users may be internal employees, partners, or clients. However they are often used repetitively every day, perhaps dozens or hundreds of times per hour or day. While users might be less sensitive to waiting a few seconds or minutes for the startup, they are highly sensitive to delays in completing their repetitive tasks, since they can greatly inhibit both individual and organizational efficiency. Think of a health care delivery app where every minute spent by a nurse or doctor impacts patient care. Or an application supporting clients where delays can impact customer wait time or the number of required support staff.

While some applications can have a mix of these characteristics, this framework can help you to better assess your approach to modernization for each application.

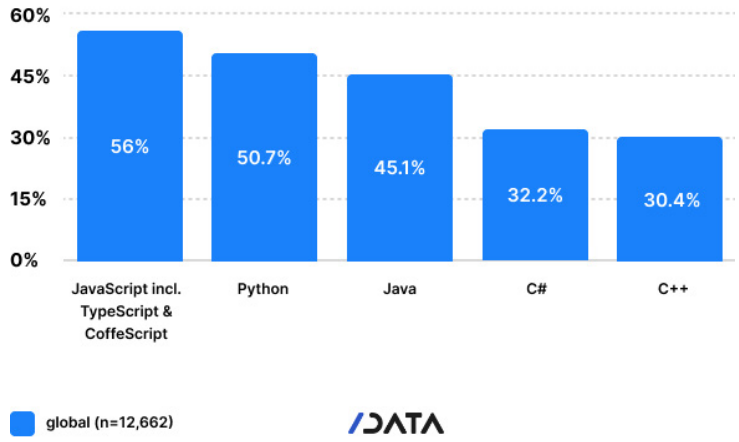
Application characteristics

	CONSUMER-SCALE	ENTERPRISE-SCALE
User volume	Millions of users	Up to thousands of users
Optimization scenario	Occasional use	Repetitive use
Performance metrics	Fast startup	Minimize task time
Goal	Simplicity	Efficiency
Example applications	Shopping, banking, mobile payments, social apps	Client support, health care delivery, data analysis, manufacturing scheduling

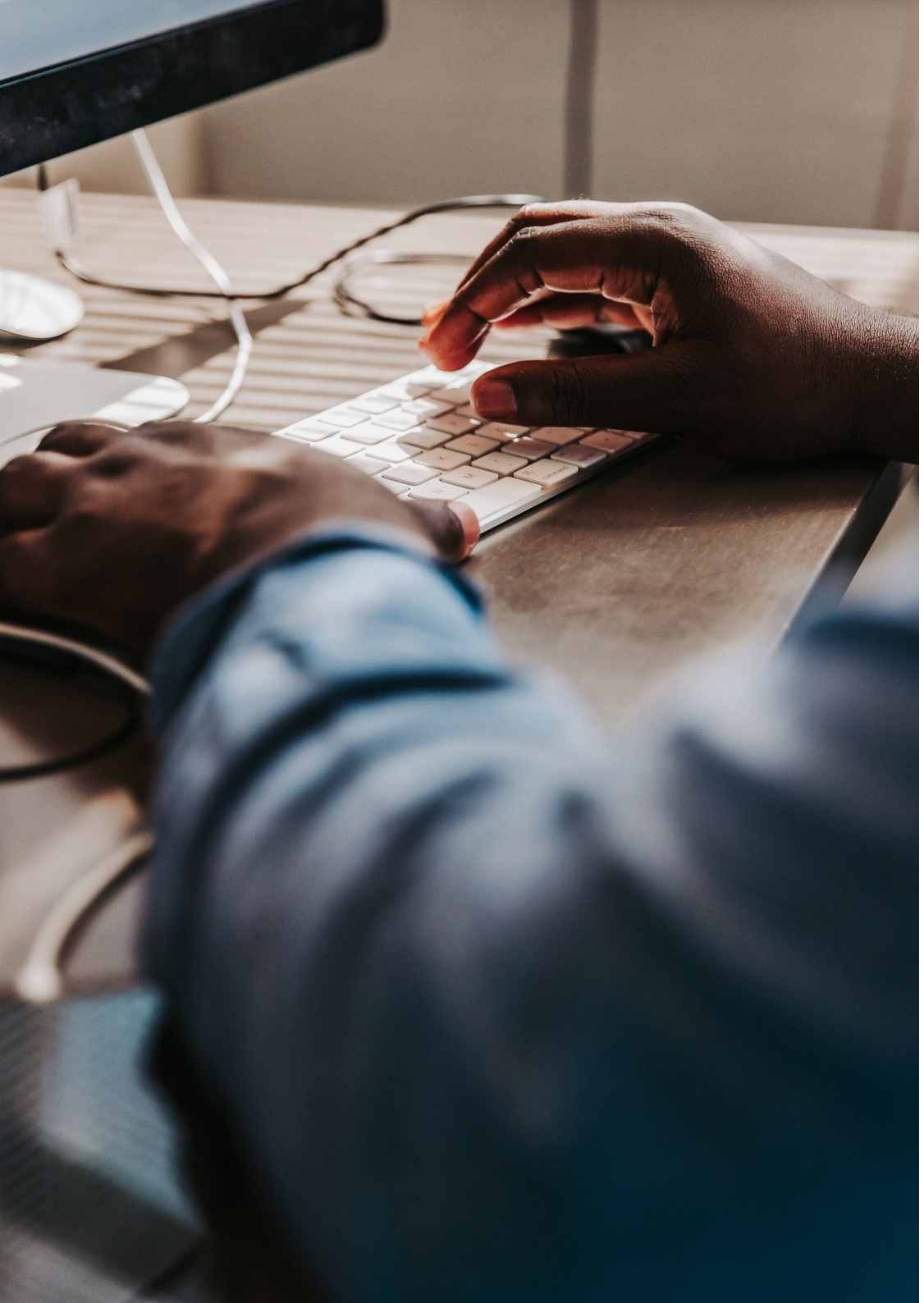
Assessing team characteristics

Another important factor to consider is the technical skill set and preferences of your available development resources. Even if you are using contractors or outsourcing work, you will still want to understand the available skill set.

In the Q1 2022 Developer Nation Pulse Report, Java and JavaScript (including TypeScript) are two of the three most popular languages. However, many developers are experts in either of these two languages. Not both.



For legacy Java applications, especially those currently running on the desktop instead of the browser, your development team may be Java experts but have limited experience in JavaScript. Even if you have hired newer developers or those that have worked on modern “consumer-scale” web applications, you will likely find that they are either back-end engineers (often using Java) or front-end engineers often with experience in JavaScript and related frameworks. In some cases, you may already have in place separate front-end and back-end teams with the requisite skill set.



Team characteristics

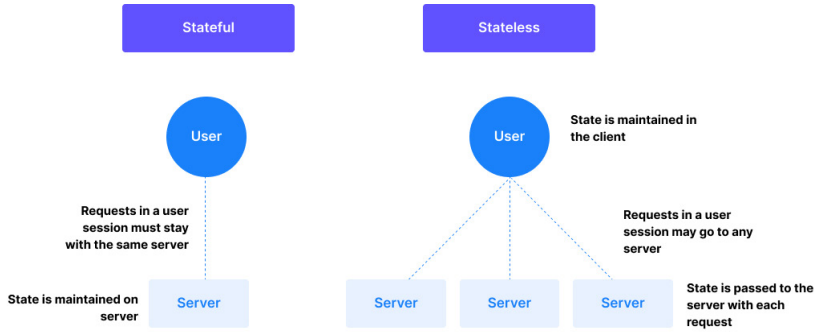
Skills	Pure Java expertise	JavaScript plus Java expertise
	Prefer to leverage Java skills without learning front-end JavaScript or frameworks	Prefer a hybrid approach with Java back-end and JavaScript for front-end

Assessing architecture approaches

Once you have assessed the characteristics of your application and your team, you can begin to assess your architectural choices. One of the key elements to consider is whether your application uses a “stateless” or “stateful” architecture. This choice is influenced by the characteristics of the application (“consumer-scale” vs “enterprise-scale”) as well as the skills of the development team. In the case of application modernization, you need to consider the architecture of the existing application as well as your desired goals.

The choice of architecture impacts the full cycle from development to deployment of a web application. Since there are pros and cons to both approaches, it is important to ensure that your choice aligns with your application characteristics and modernization goals.

Understanding stateful vs stateless applications



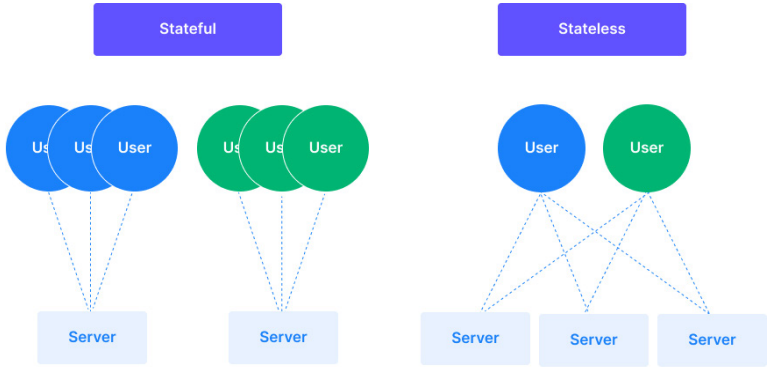
In computer science, “state” is a broad term used to describe the current value and content of an application. In web applications, it refers to maintaining sessions for authentication and similar processes.

	Stateful application	Stateless application
Definition	“Each client initiates a session on the server and then invokes a series of services on the server, finally exiting the session. **Application state is kept entirely on the server.”** Source: Roy Thomas Fielding PhD Dissertation	“Each request from client to server must contain all of the information necessary to understand the request, and cannot take advantage of any stored context on the server. **Session state is therefore kept entirely on the client.”** Source: Roy Thomas Fielding PhD Dissertation
Server interaction	Requests during a user session must go to **the same server** that holds their state data – or they lose the context of previous transactions.	Requests during a user session may go to **any server** since all of the “state” information from that session is held in the client and passed with each request.

Assessing approaches to scalability

With the rise in cloud computing, stateless applications have become more common, especially for consumer-scale applications. These approaches make it easier to scale applications up and down “horizontally” to meet user demands by simply adding and removing virtual machines in a cloud environment. Think about the online shopping site that needs to serve more users in the evening or on weekends, or scale up to meet huge demand during holiday shopping surges. With a stateless architecture, scaling infrastructure down as user load goes down doesn’t interrupt any current user shopping sessions. The user session can continue on any of the remaining VMs.

Conversely, because stateful applications require communications from a particular user session to continue with the same server, it can be challenging to use the horizontal scaling methods popularized by cloud and Kubernetes technology. In these models, as the back-end service scales down, it may shut down a CPU that is in the middle of serving a user session, thereby losing the user’s state. While there are potential workarounds to these issues, developers must consider how to handle these situations if using horizontal scaling for a stateful application.



Assessing other technical requirements

In addition to application architecture and scalability, other factors influence your approach to Java application modernization.

Startup time and performance

For ecommerce applications, such as web stores, startup time is key. Not only are buyers more likely to find the application since it ranks higher in the Google search results, they are also less likely to commit to a purchase if the online store is perceived as slow.

Startup time matters less on complex applications, such as business applications, where users are expected to stay on the apps for hours. Instead, it's more important that the time to complete a task is as short as possible. This is especially critical in applications where users must complete particular tasks, dozens or hundreds of times a day.

Cross-field validation

Cross-field validation entails validating that data is correct by comparing it to another field. For example, a password check must be validated in conjunction with a user name. This validation is especially critical when working with data that has multiple input values, for example, in complex banking transactions.

Initial validation for UX purposes can be done on the client. However, final validation must be done where the state can be trusted, which is on the server. For stateless applications, this means you'll have to rebuild the state and re-validate on the server for data validity purposes.

Security

The stateful versus stateless approach is also a consideration when addressing security. For example, in a stateful application, it's easier to authenticate, because everything the authentication requires is kept in the session state on the server. In a stateless application, you may need to leverage more complex techniques such as secure tokens to avoid relying on the "untrustworthy" browser to tell you who the user is.

Logging customers also requires different implementation techniques. The stateful approach is straightforward — you just authenticate once at login and know throughout the whole session who is at the other end. In contrast, token-based authentication used for stateless apps is quite a recent approach and requires a fair amount of learning new concepts. Implementing a command such as "throw out user X from the system" can turn out to be surprisingly difficult in a stateless setup.

Support availability

No one knows the software as well as the people who built it. An important consideration for many enterprises is the availability of vendor-backed support, where the software provider is committed to providing expert support for utilizing their software. Support options include training, expert development support, and "hot-lines" when something goes wrong.

There are also 3rd-party companies that can provide the necessary support for your business. In either case, scoping the need and availability of support is key when committing to a technology-stack.

Mapping your path with Vaadin

Vaadin provides two proven frameworks for delivering Java-based web applications with a modern UX and productive DX.

Vaadin Flow

Vaadin Flow is the only framework that enables developers to build full-stack web applications with a modern, intuitive UI completely in Java. It operates a secure, server-side architecture perfectly suited for stateful applications.

Hilla from Vaadin

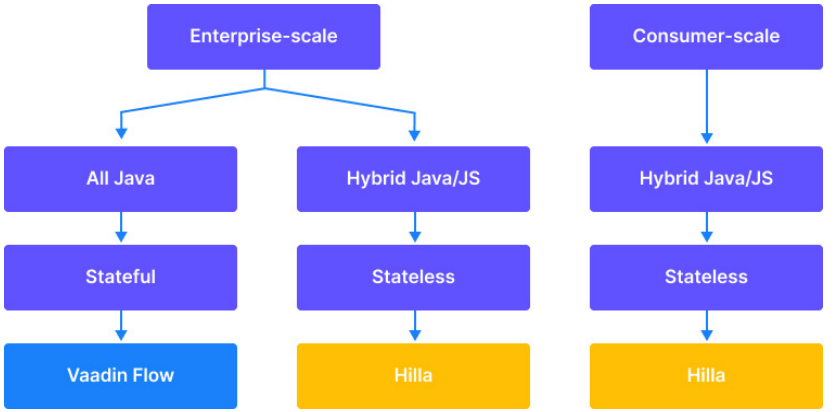
Hilla provides a faster and easier way to integrate a reactive TypeScript front-end with a Java back-end. Hilla integrates seamlessly with SpringSecurity for secure endpoints and supports stateless architecture and horizontal scaling.

Pre-built UI components from Vaadin

Both Vaadin Flow and Hilla can leverage over 50 web UI components that are commonly used in business applications. These range from foundational form and input fields to complex grids and charts. Vaadin UI components greatly accelerate the time to build an intuitive UX.

Selecting the right Vaadin framework

After assessing the factors discussed above, you can choose the right Vaadin framework for your use case.



Contact Vaadin to learn more about Vaadin tools and services to help with Java application modernization.

vaadin}>