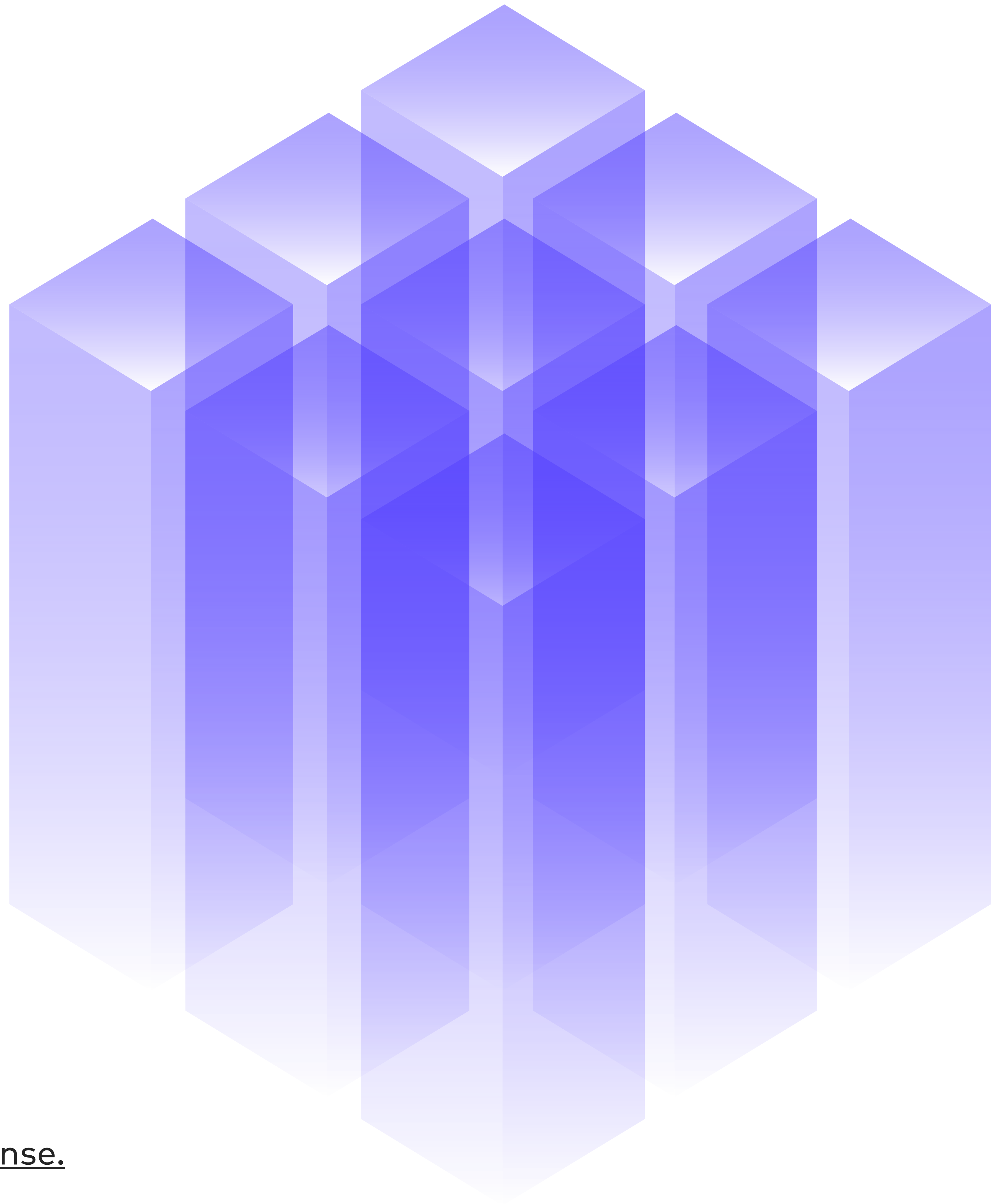
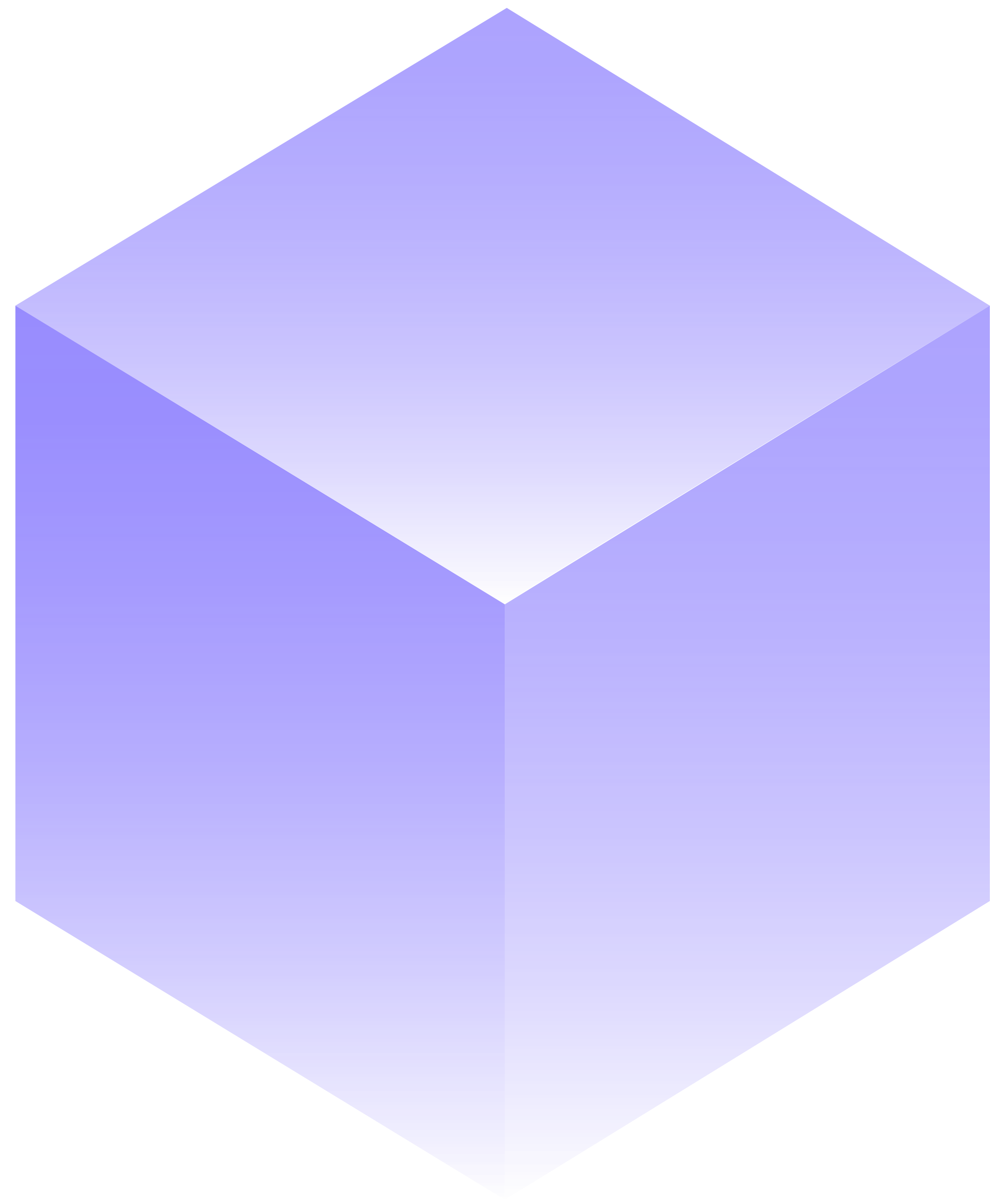
Self-contained Systems (SCS)

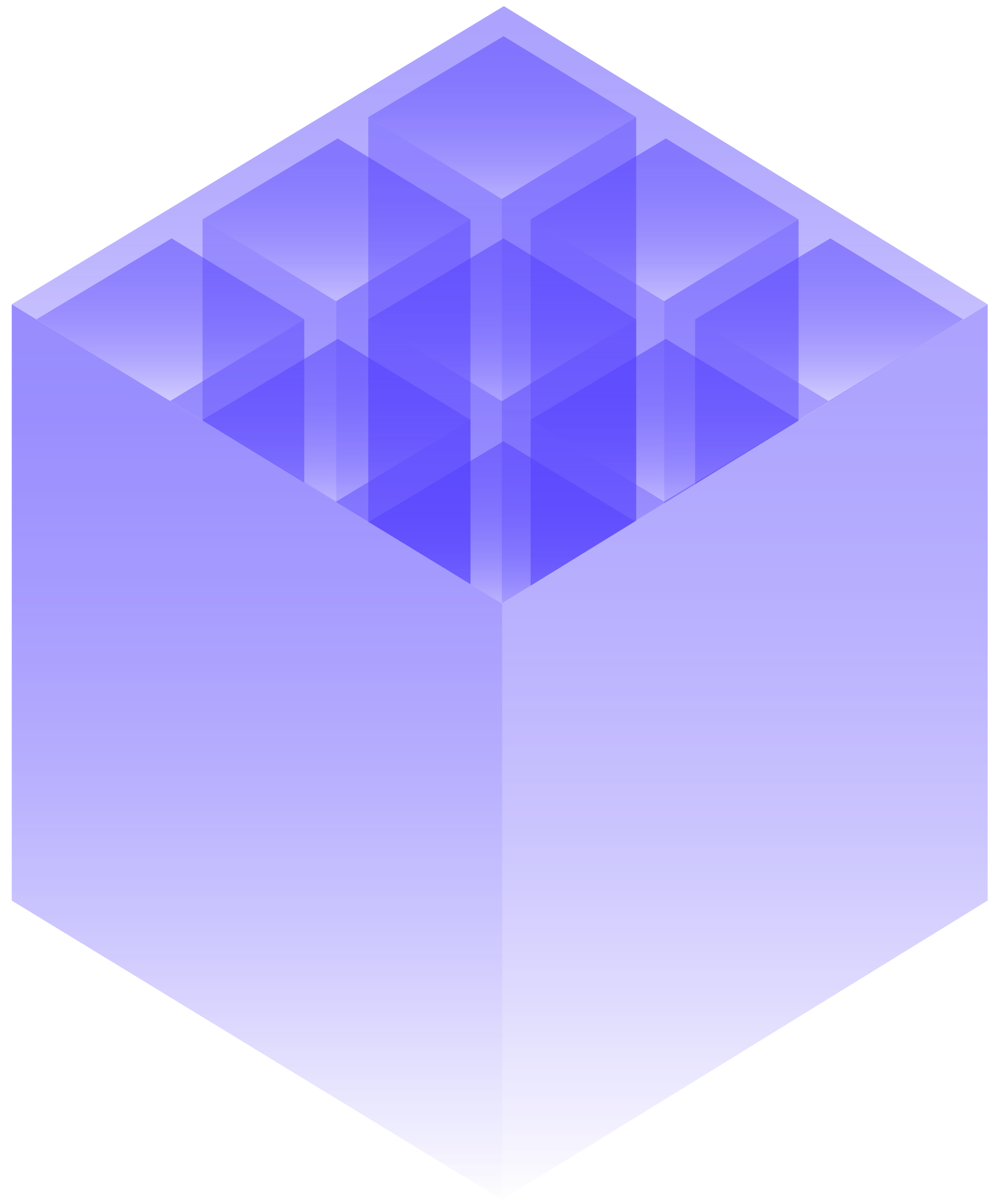
An architectural approach that separates a larger system's functionality into many independent, collaborating systems.

Created by INNOQ. Driven by the Community.

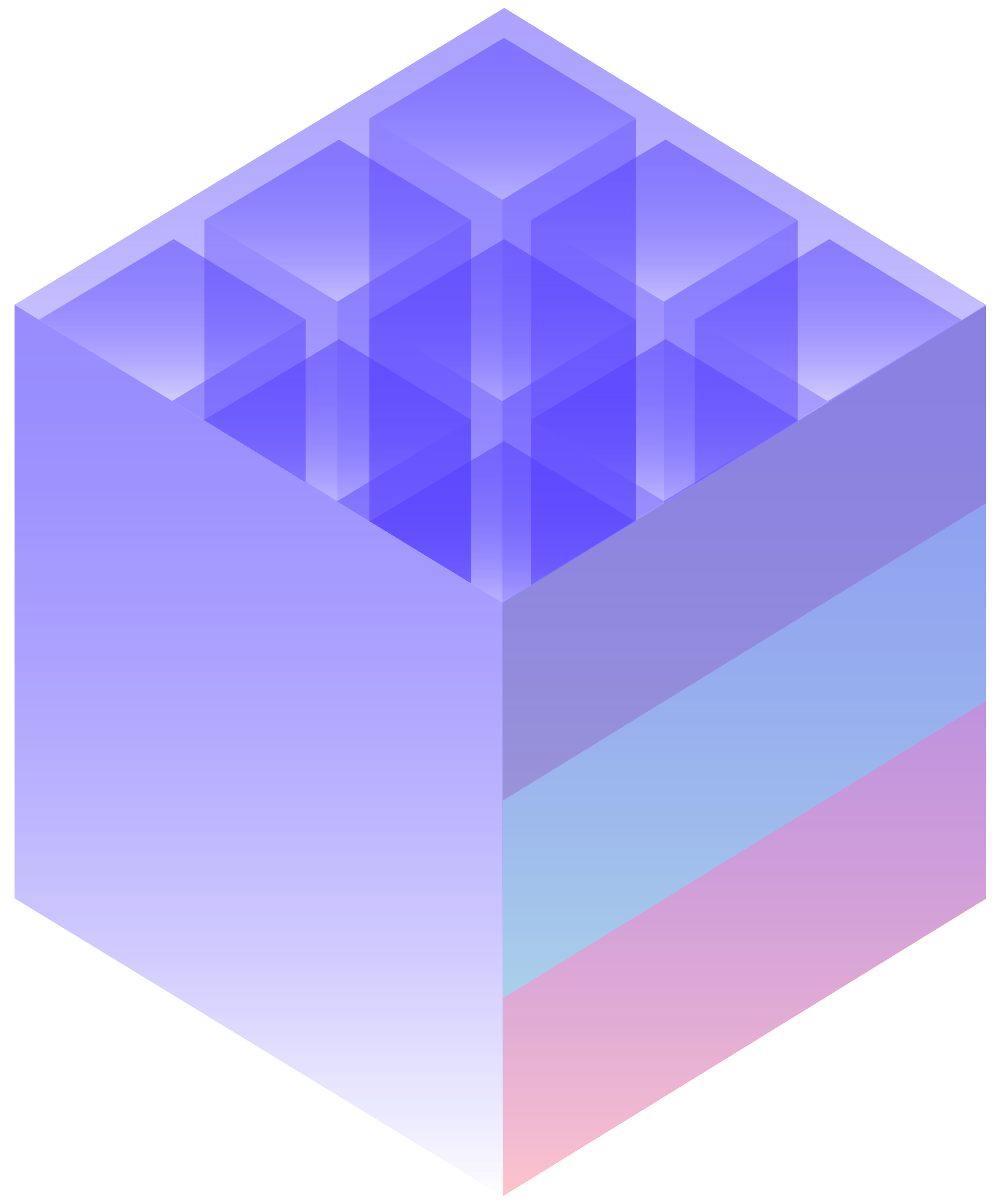




A monolith contains **numerous** things inside of a single system...

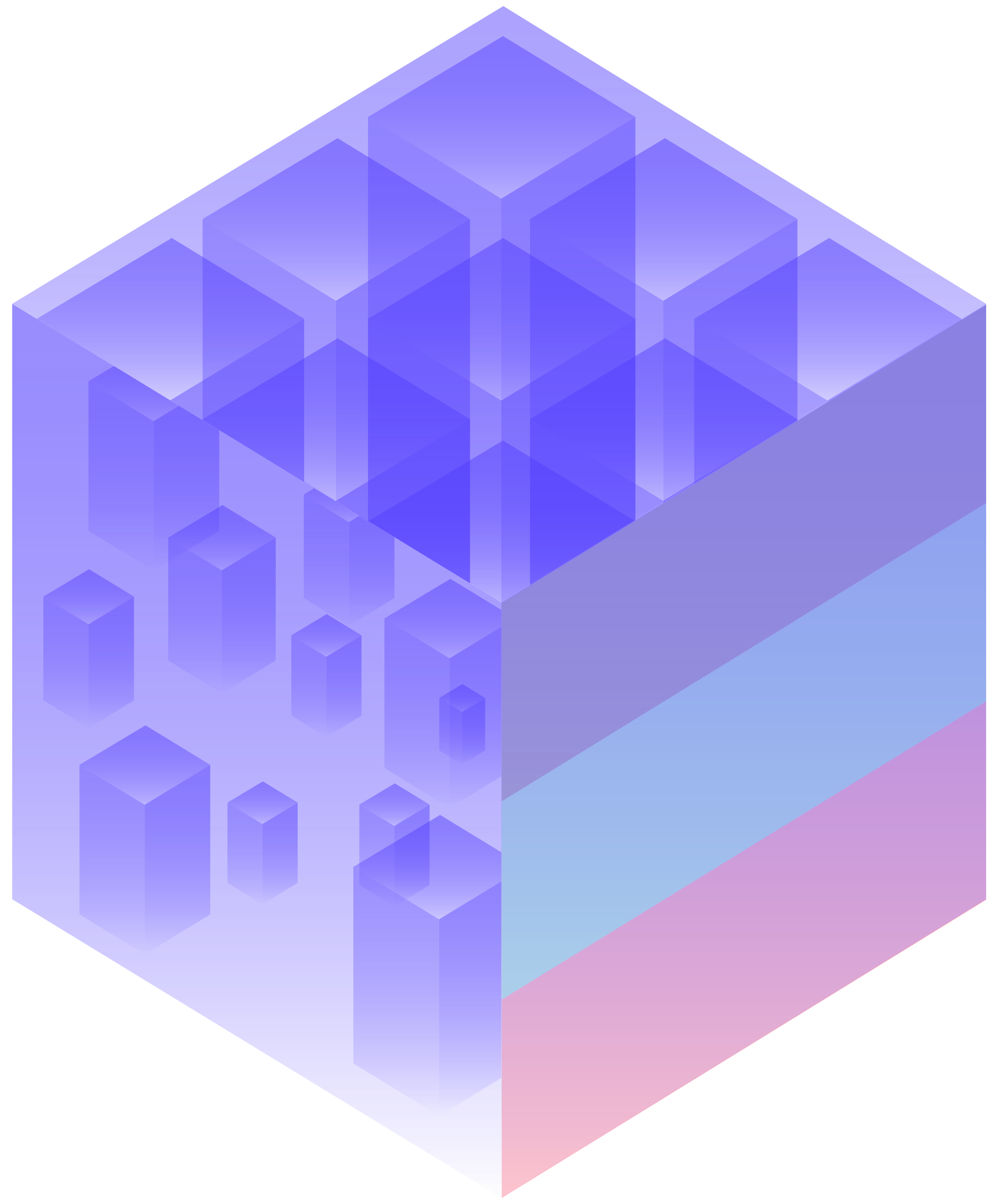


Various Domains

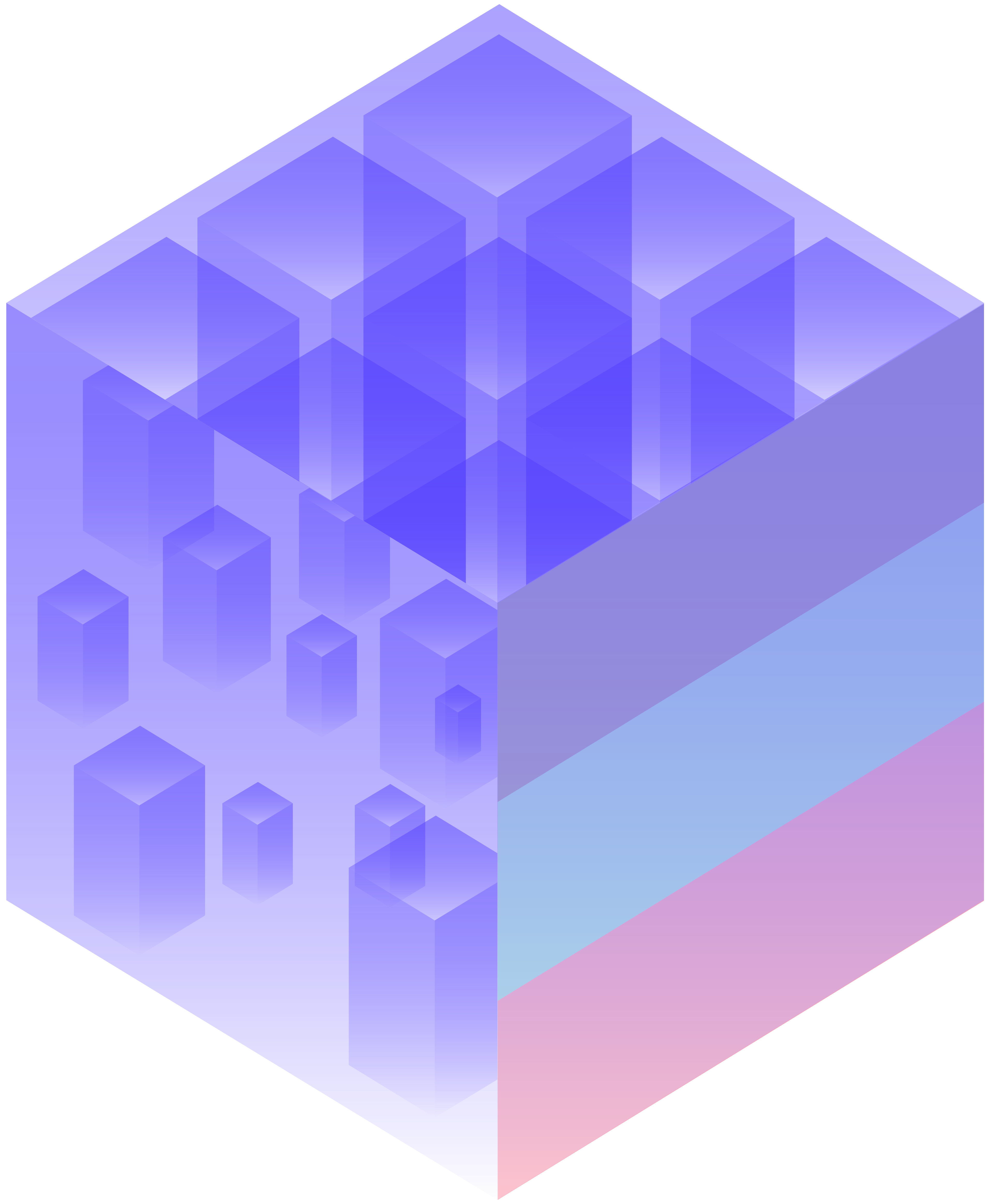


User interface
Business logic
Persistence

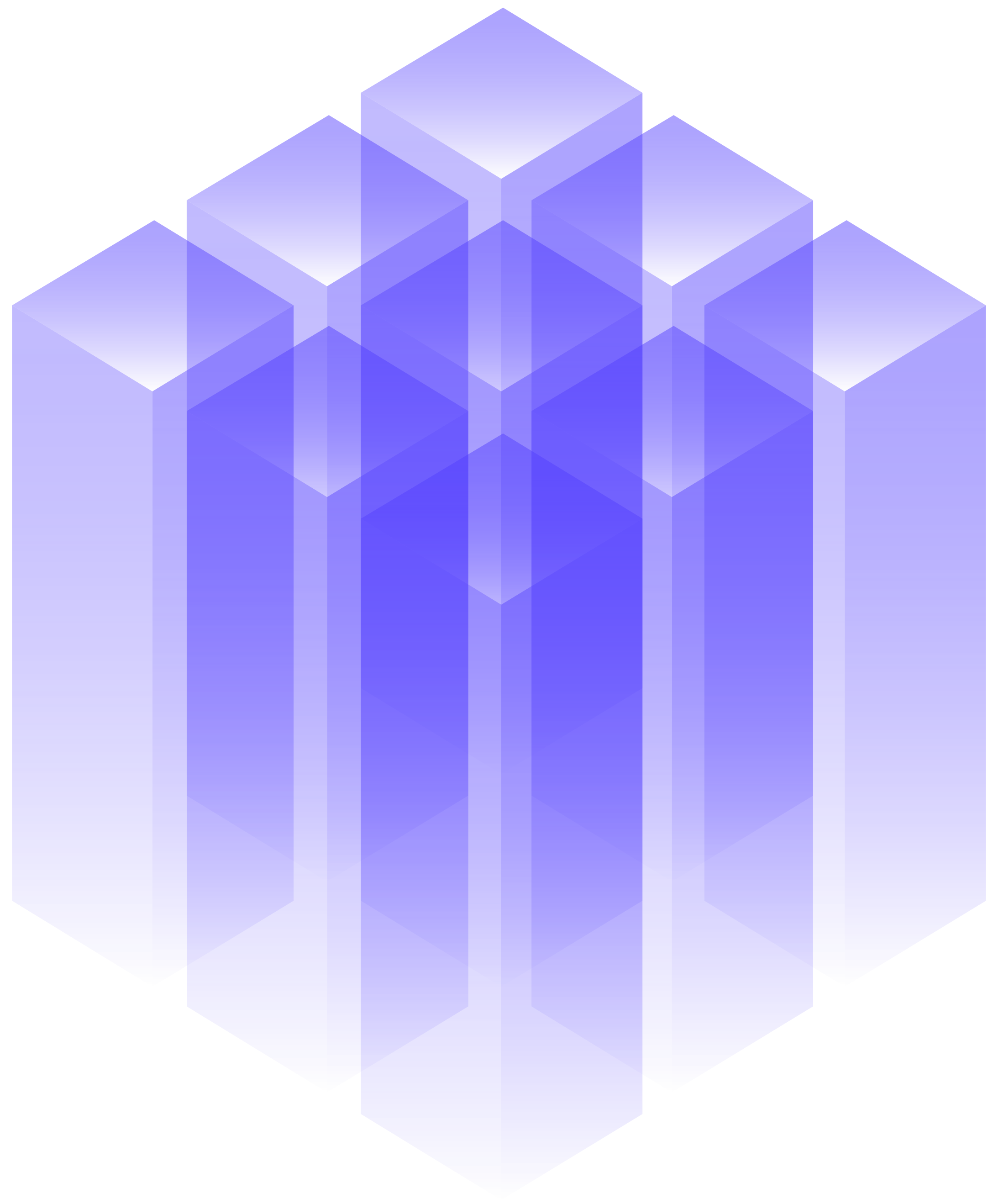
…as well as a **lot** of modules, components, frameworks, and libraries.



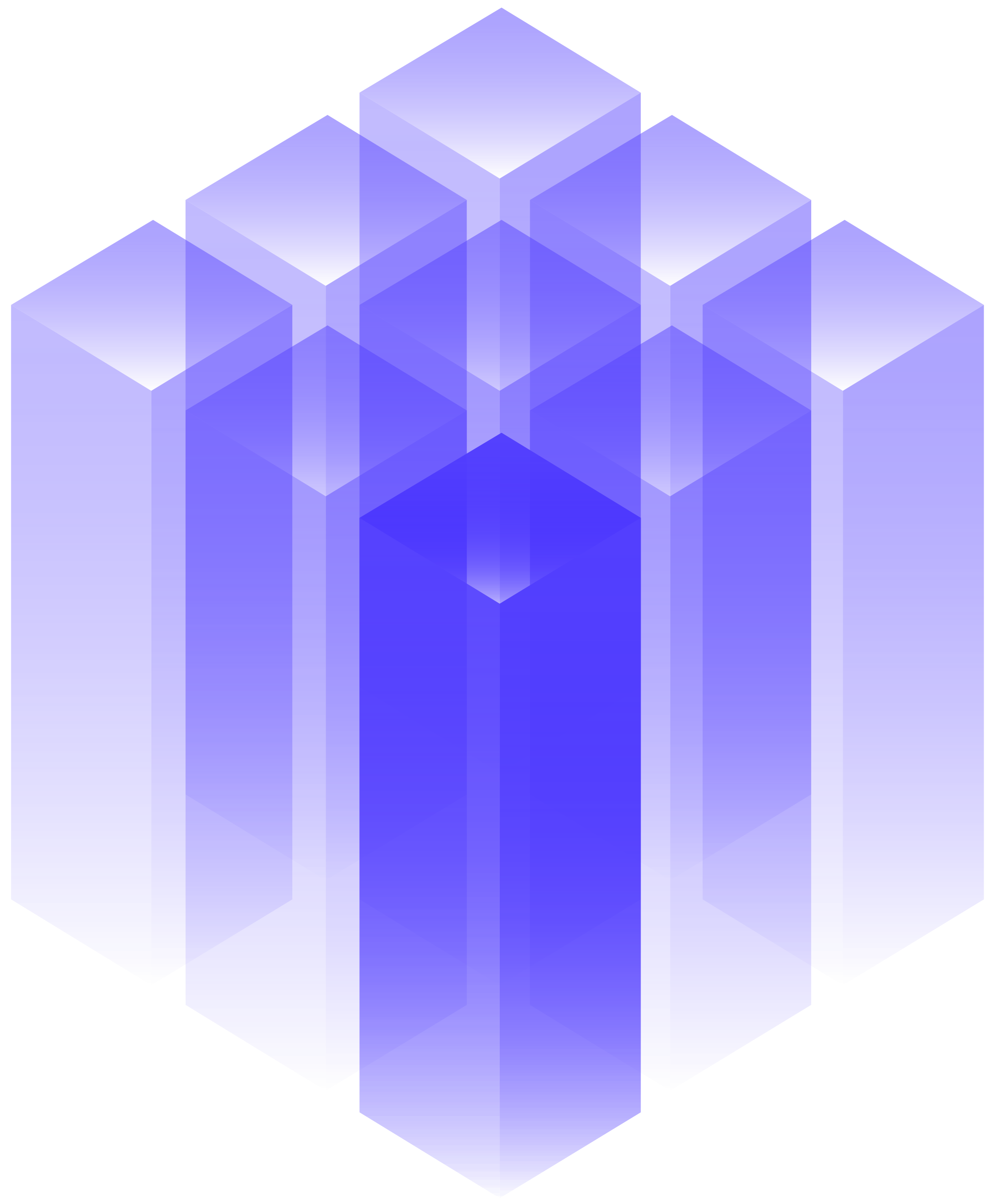
With all these layers in one place, a monolith tends to **grow.**



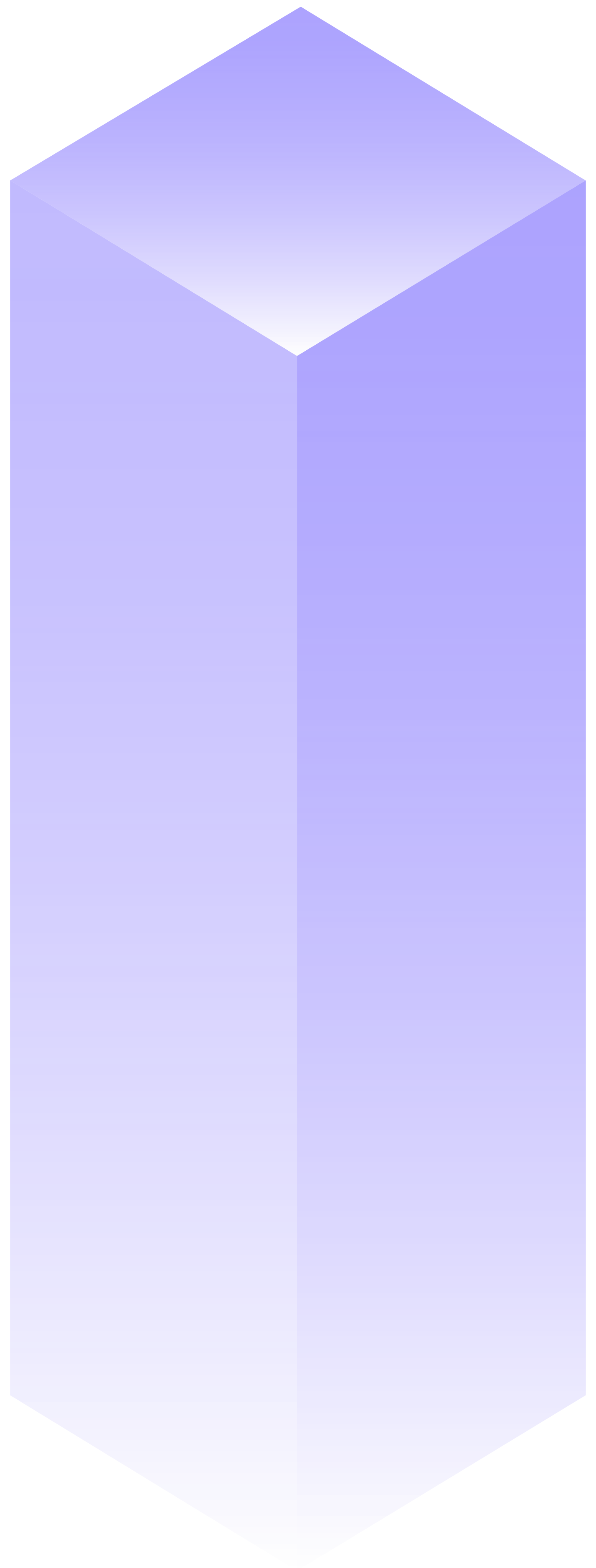
With all these layers in one place, a monolith tends to **grow.**



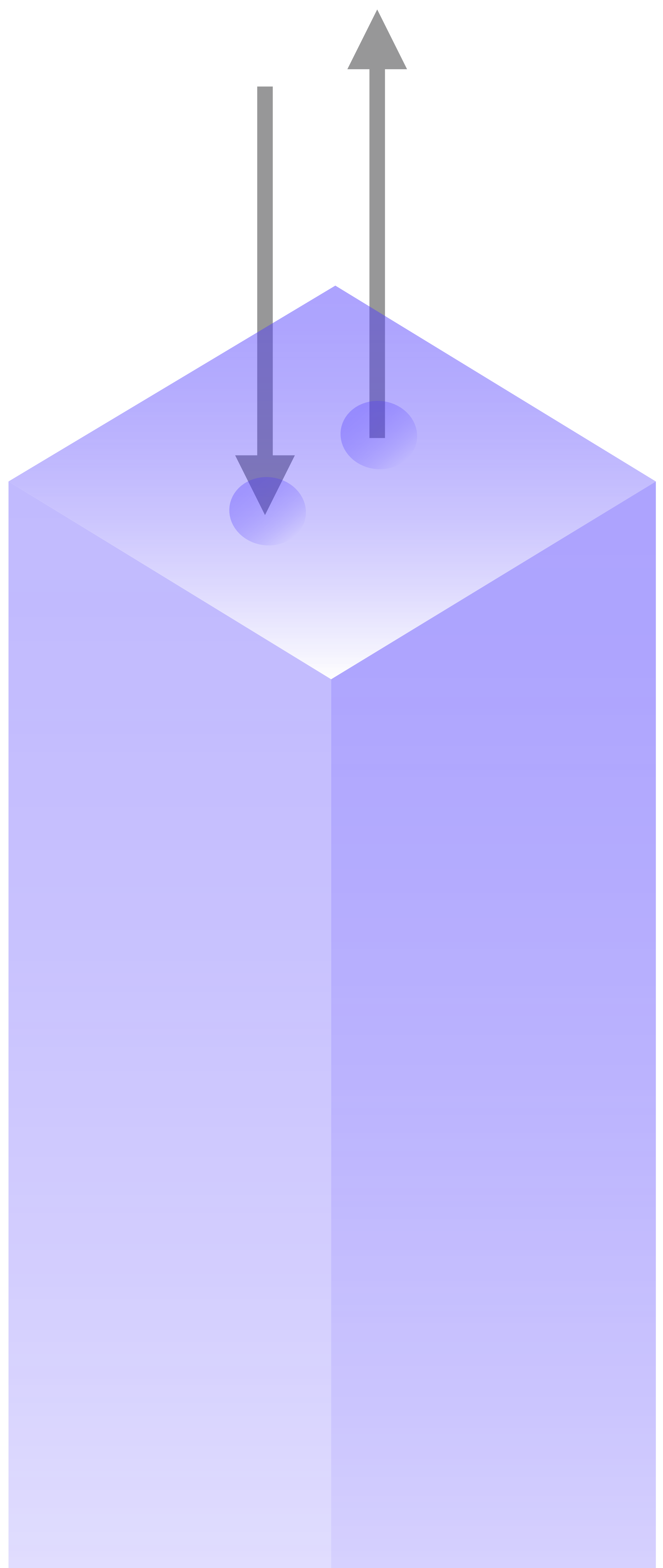
If you cut a monolithic system along its very domains...



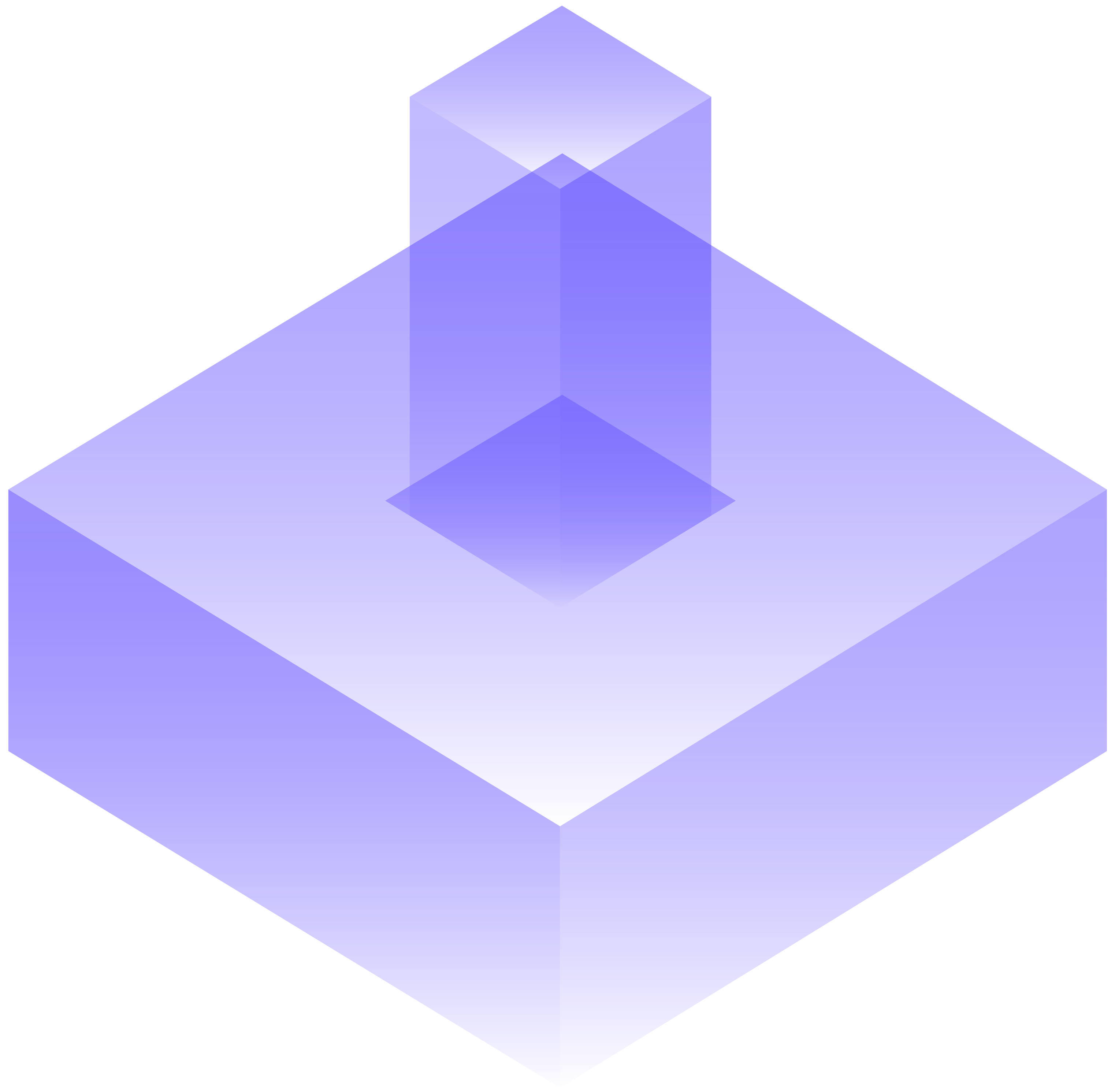
...and wrap every domain in a **separate, replaceable** web application...



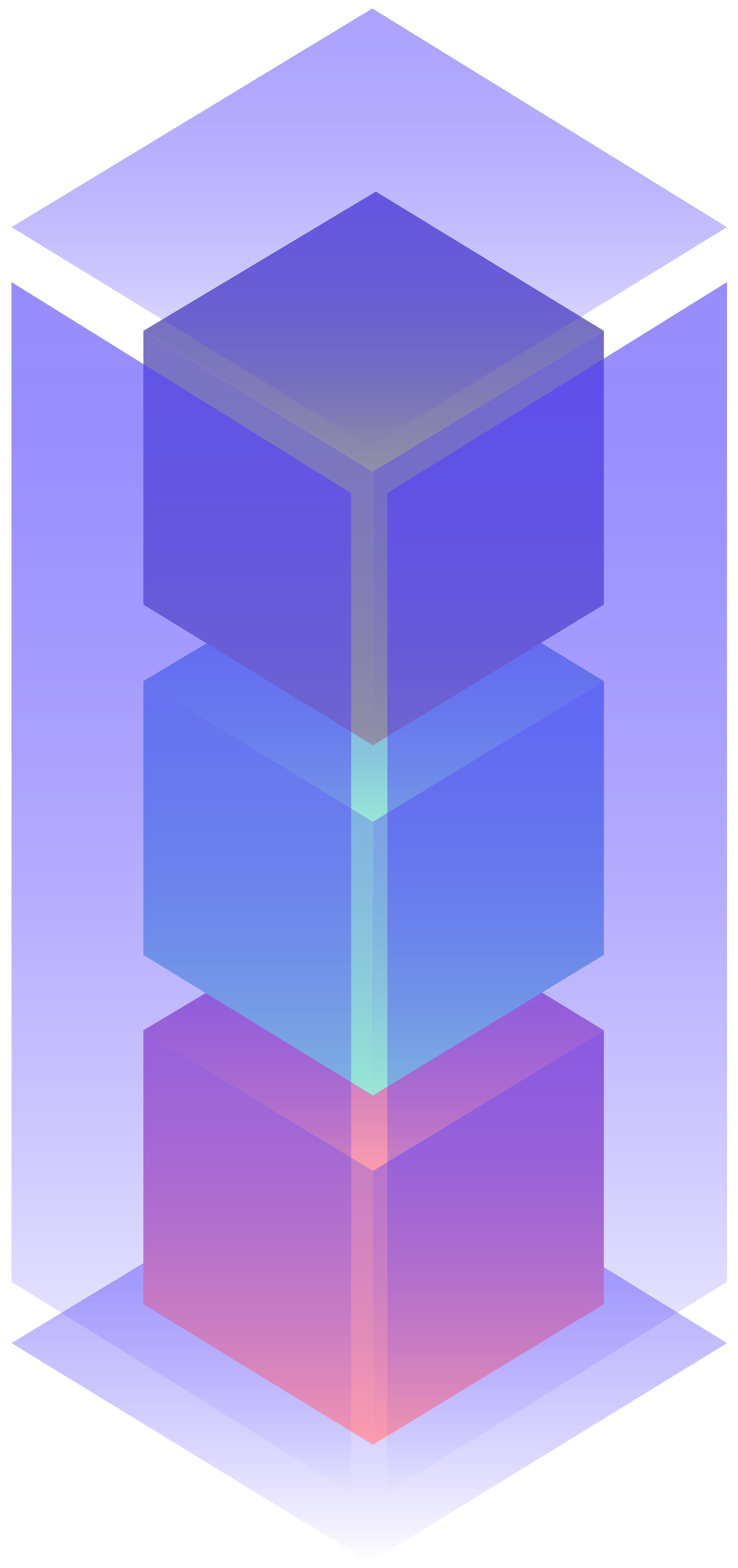
...then that application can be referred to as a **self-contained system** (SCS).



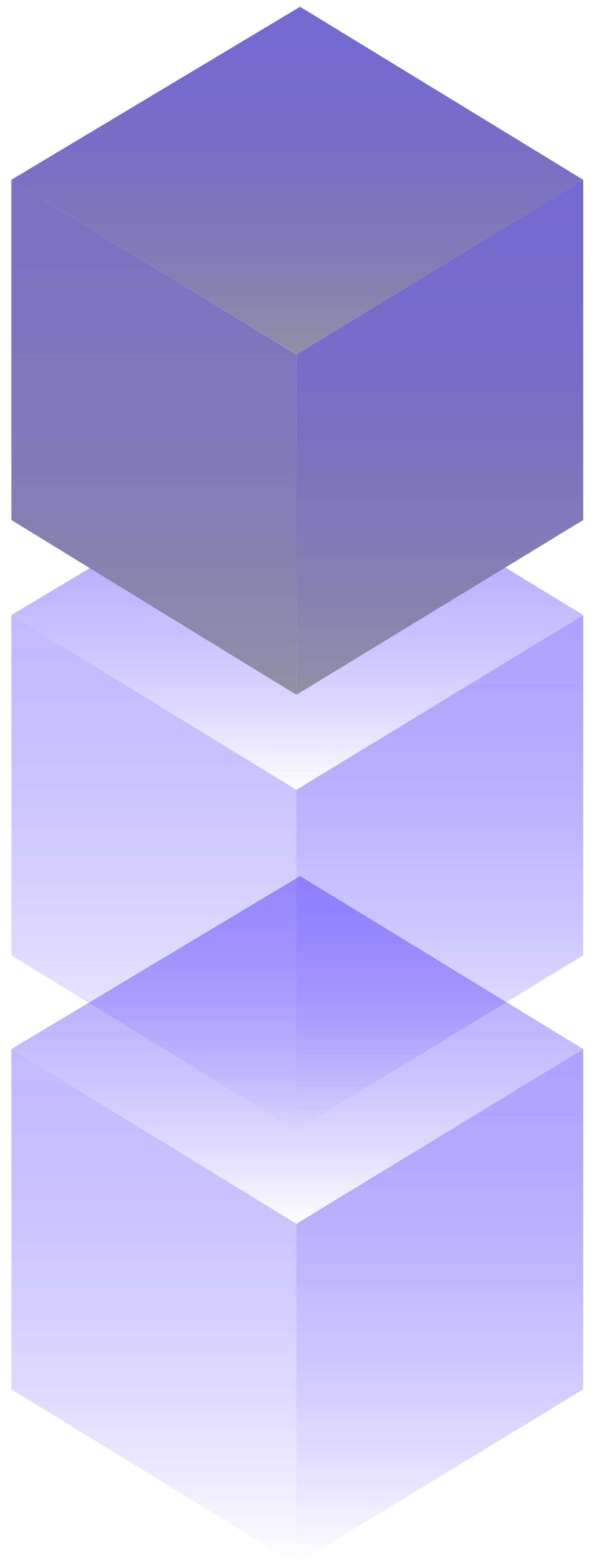
On its outside, an SCS is a decentralized unit that is communicating with other systems via **RESTful HTTP** or **lightweight messaging**.



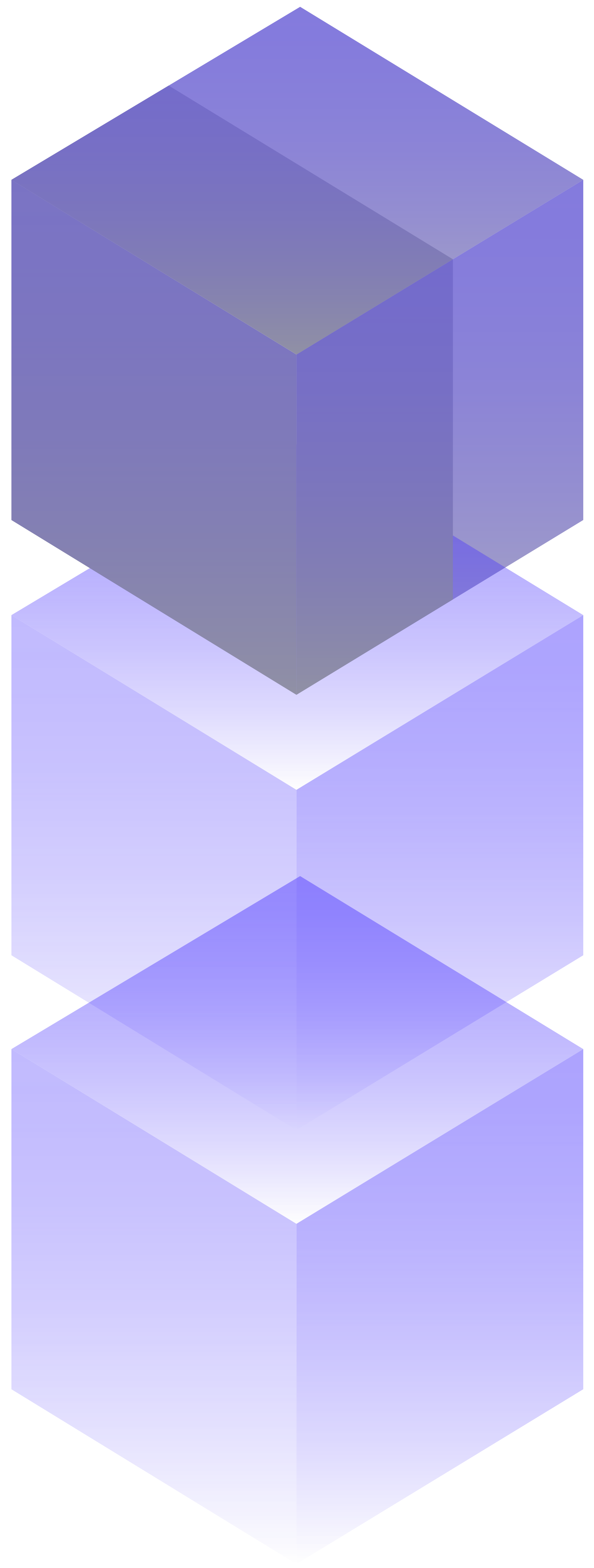
Therefore self-contained systems can be individually developed for **different platforms.**



An SCS contains its own **user interface,** specific **business logic,** and separate **data storage**



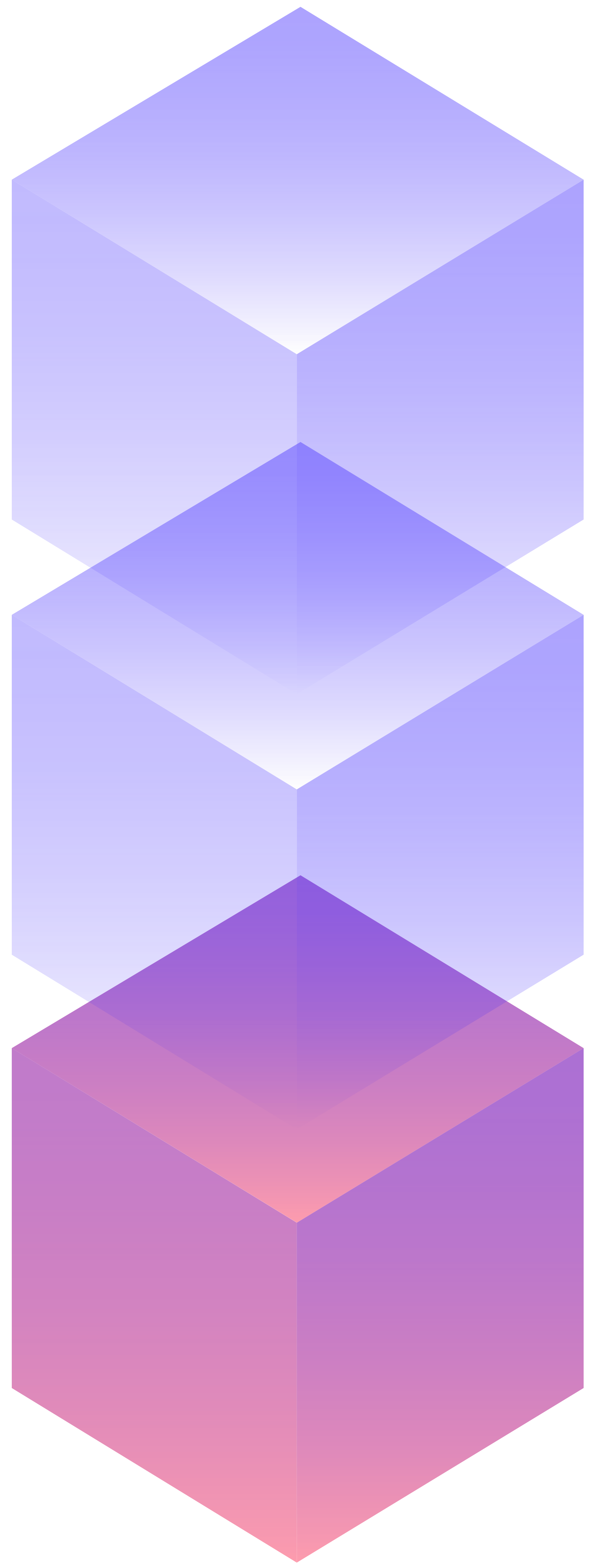
The user interface is comprised of web technologies, assembled in accordance with **ROCA principles.**



In addition to a web interface, a self-contained system can also provide an **optional API**.

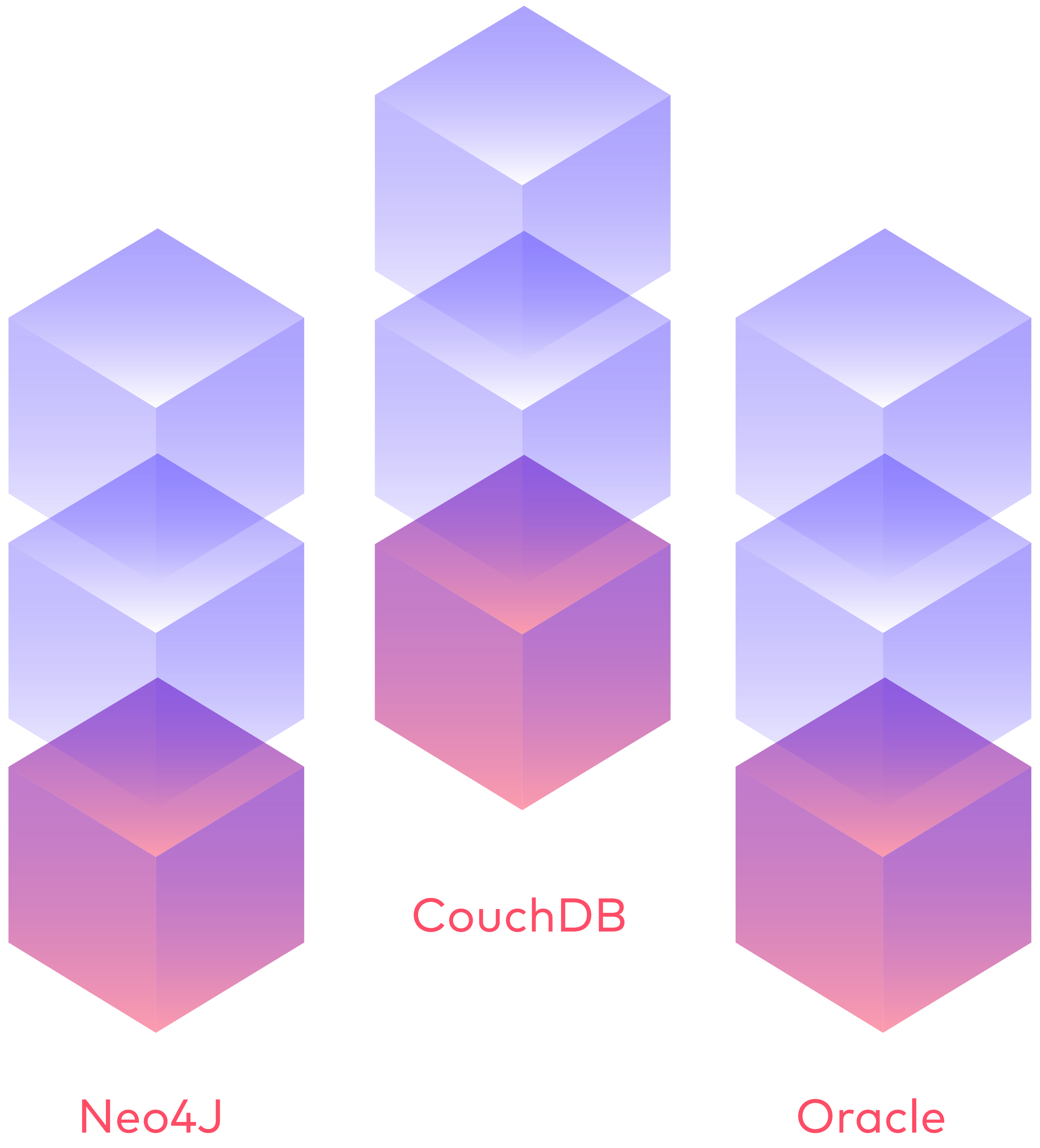
The business logic part **solely** addresses problems that occur within its core domain. This logic is shared with other systems only through a **well-defined interface**.

The business logic may comprise **microservices** designed to solve domain-specific problems.

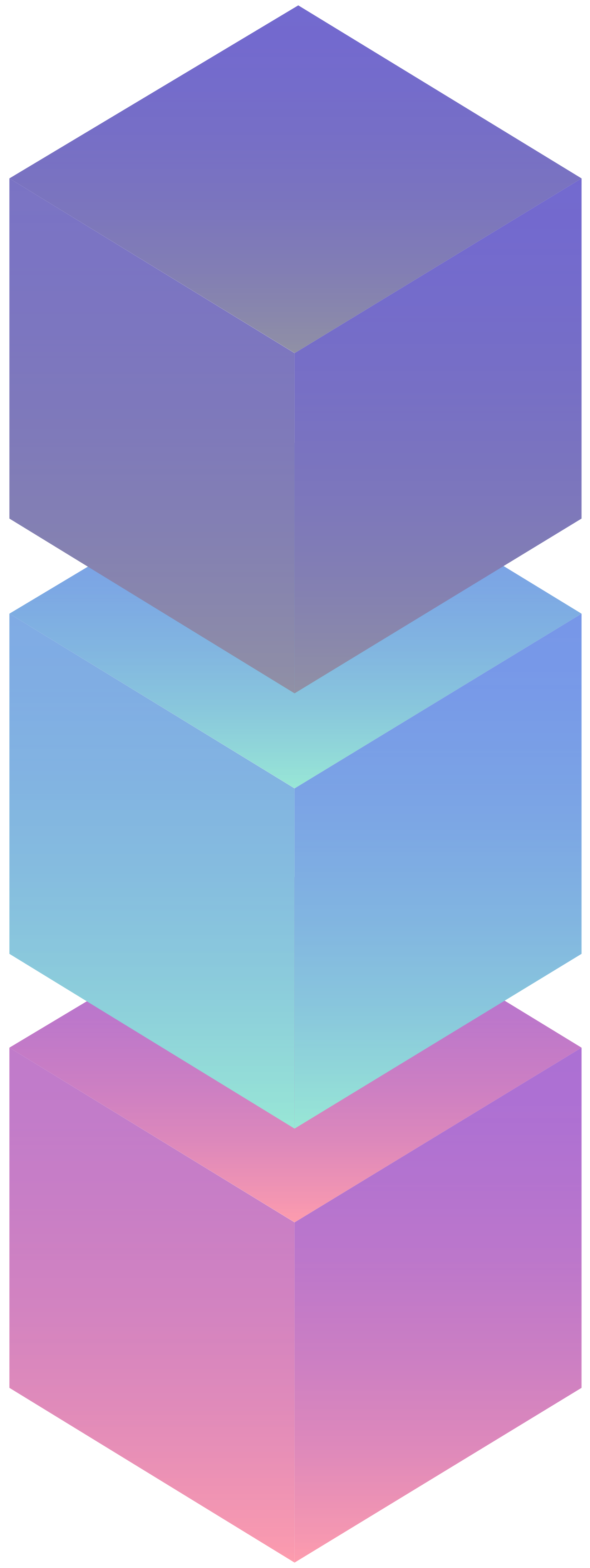


Every SCS brings its **own data storage** and with it redundant data depending on the context and domain.

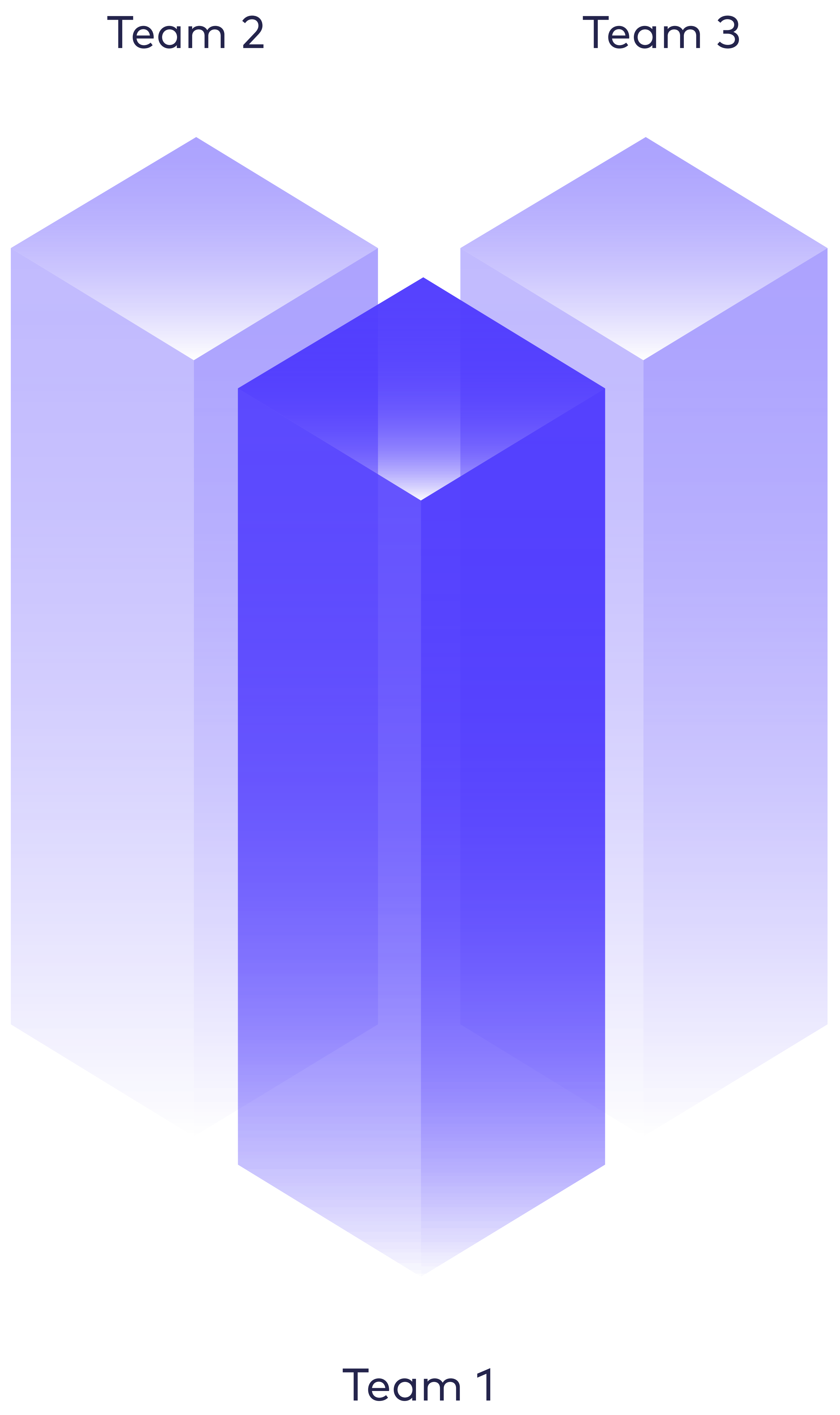
These redundancies are tolerable as long as the **sovereignty of data** by its owning system is not undermined.



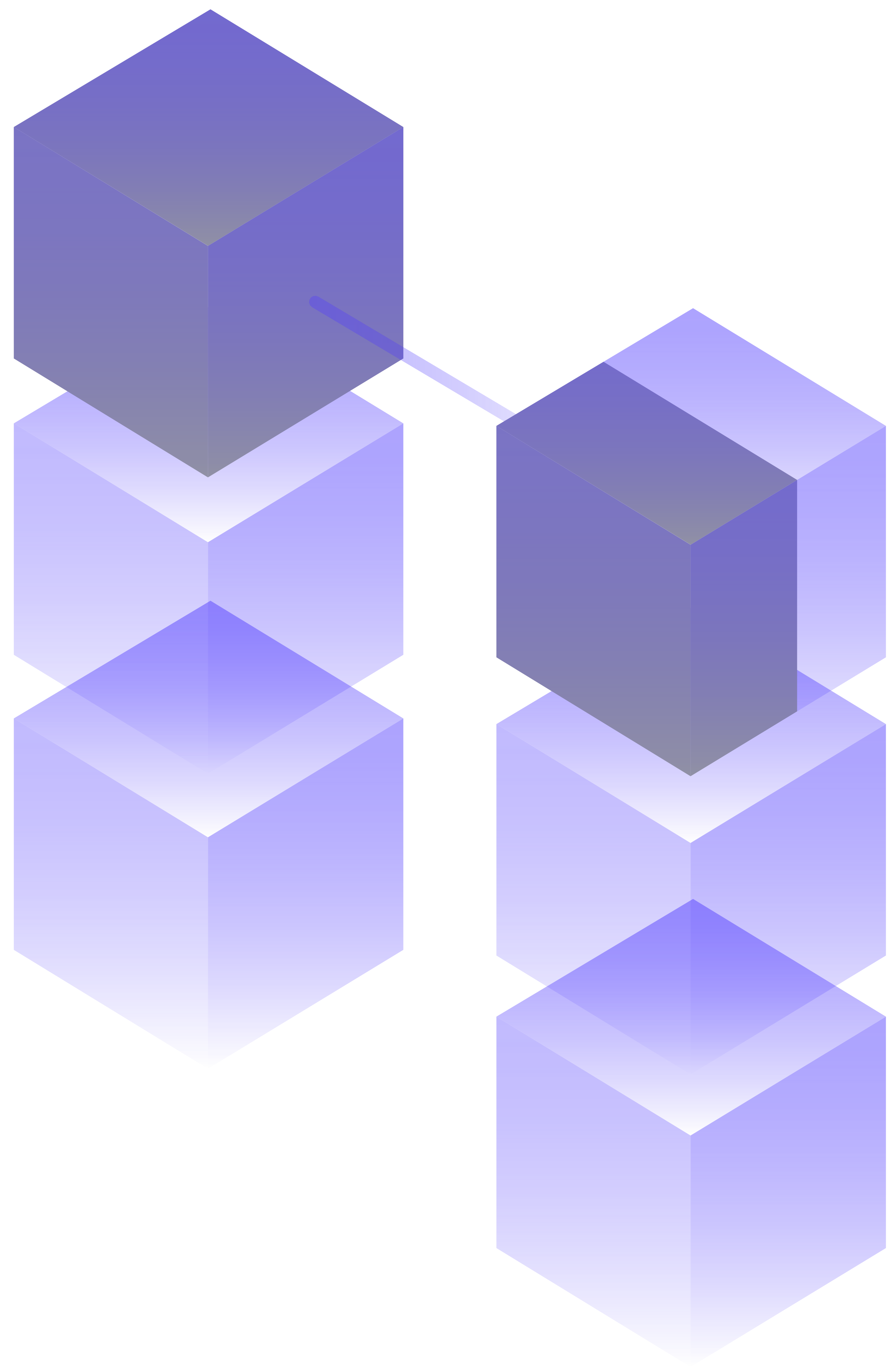
This enables **polyglot persistence**, implying that a database can be selected to solve a domain specific problem, rather than to fulfill a technical urge.



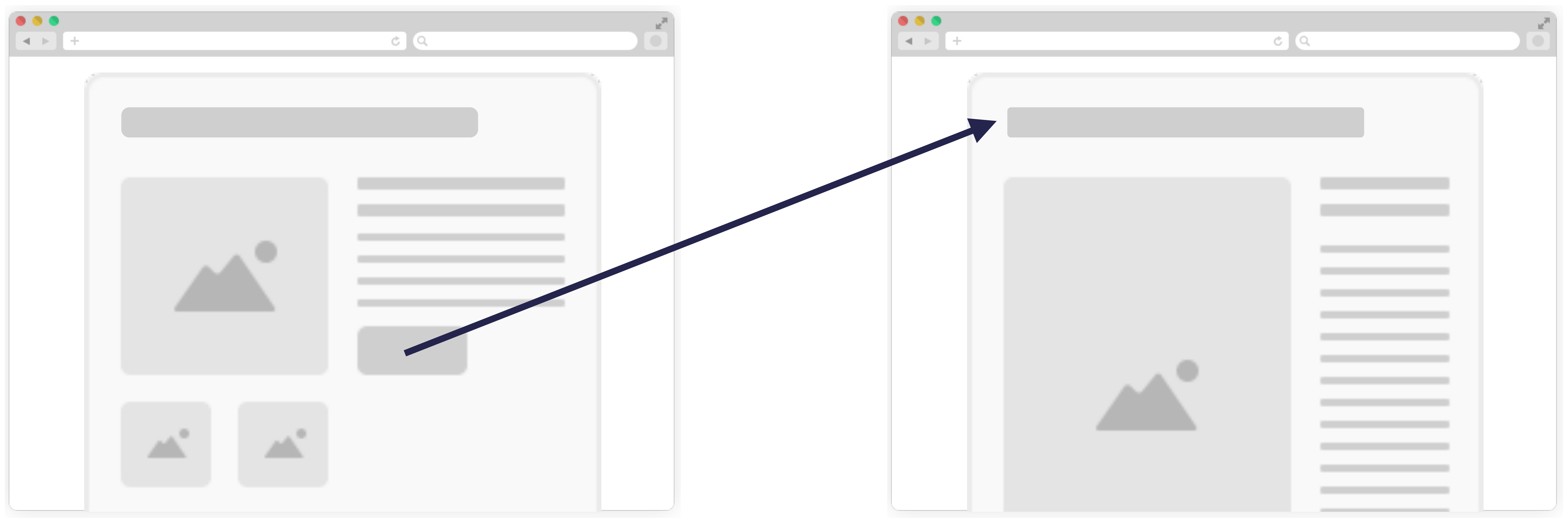
Within a self-contained system, a variety of **technical decisions** can be made independently from other systems, such as choices regarding programming language, frameworks, tooling, or workflow.



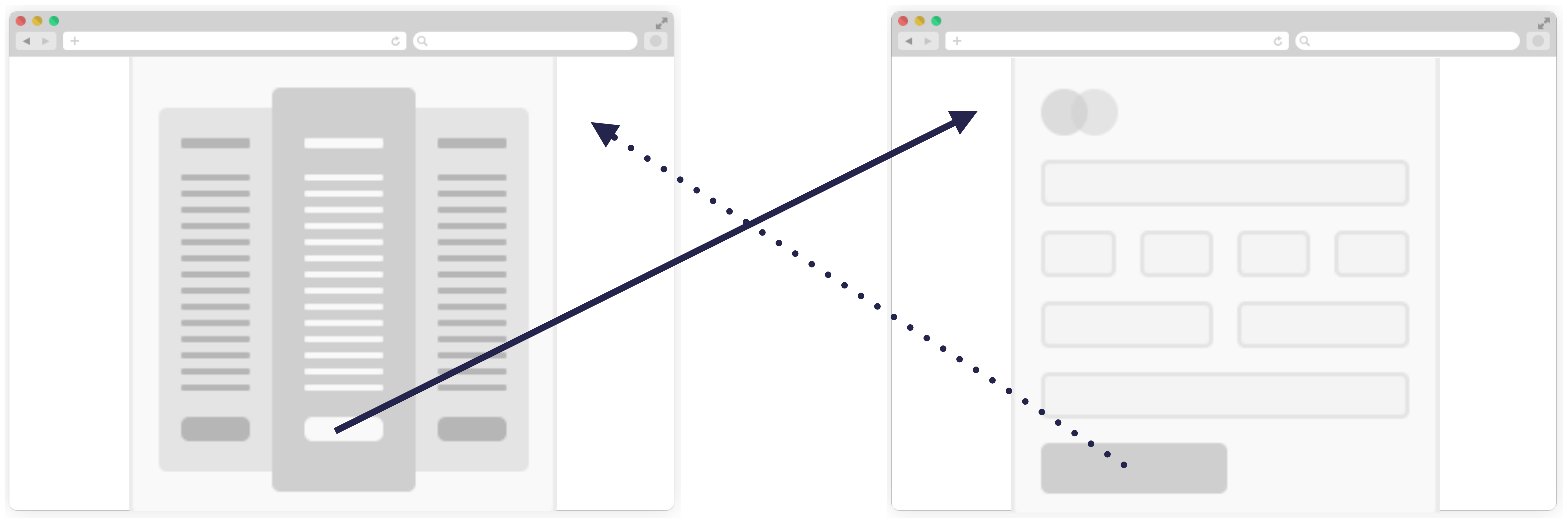
The manageable domain specific scope enables the development, operation, and maintenance of an SCS by a **single team**.



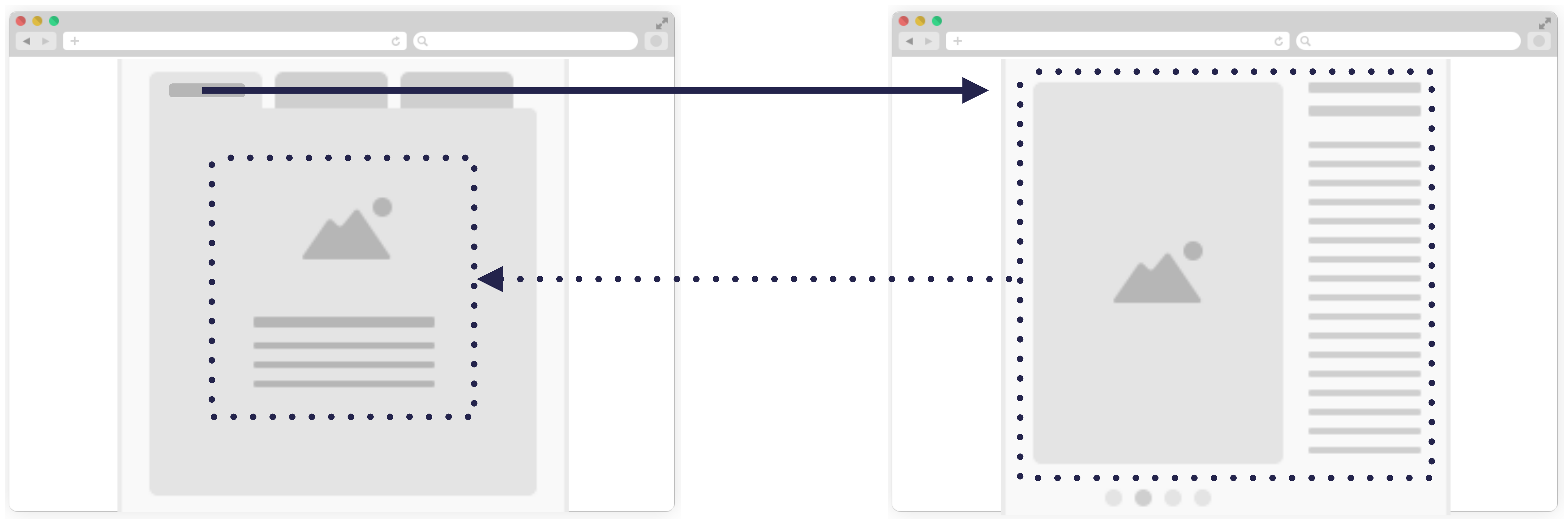
Self-contained Systems should be integrated via their **web interfaces** to minimize coupling to other systems.



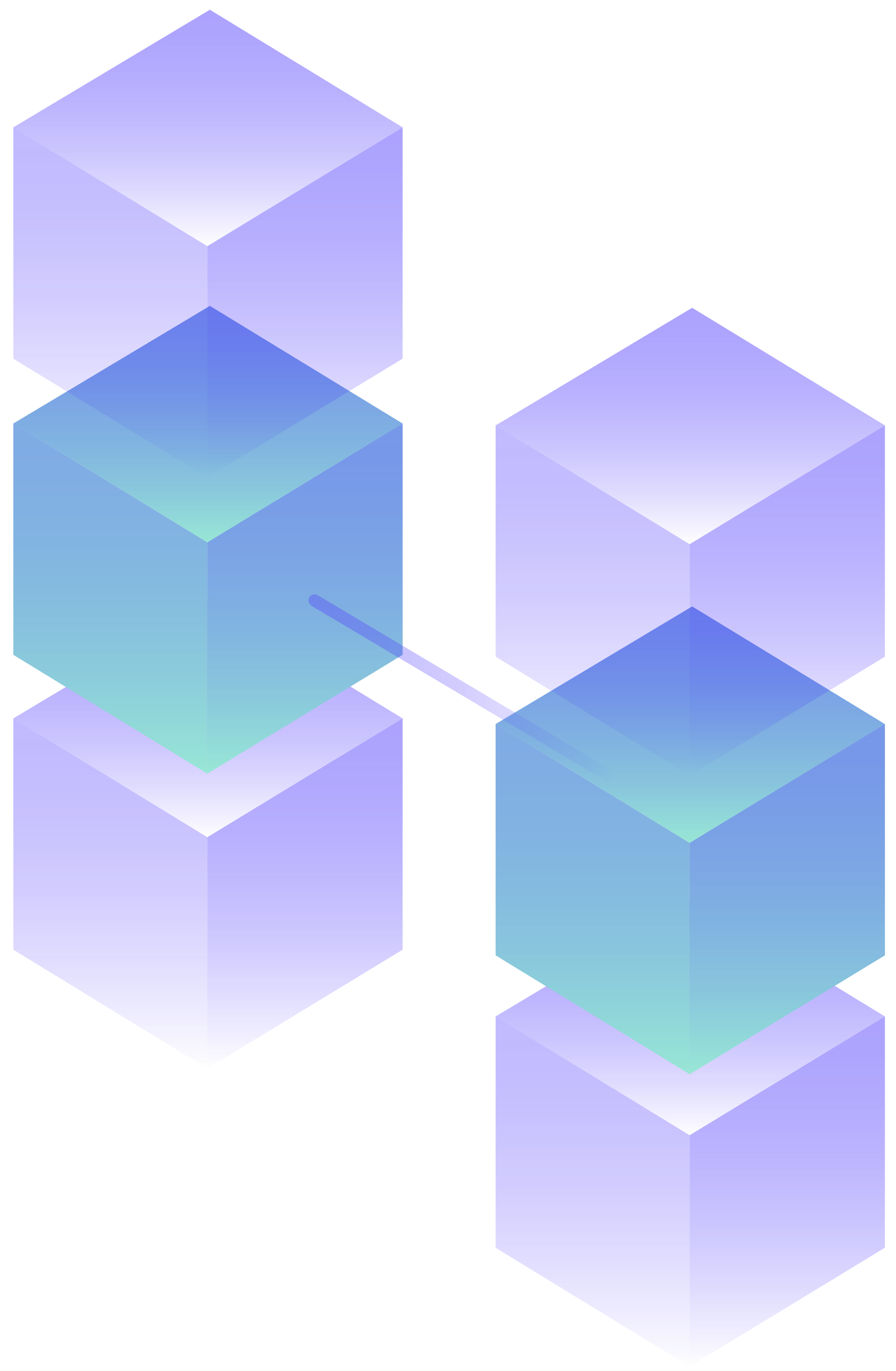
Therefore, simple **hyperlinks** can be used to navigate between systems.



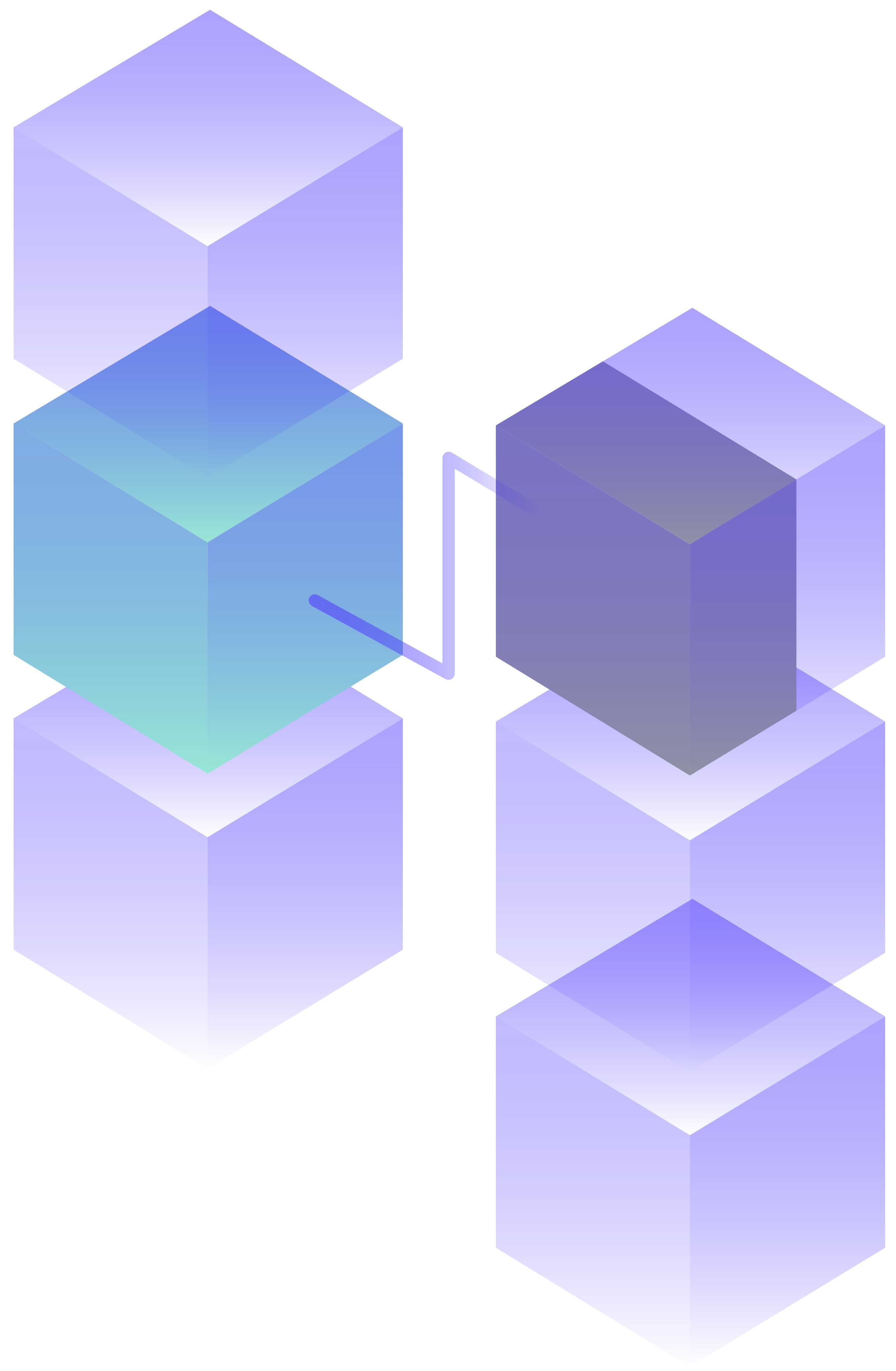
Redirection can be used to ensure navigation works in both directions.



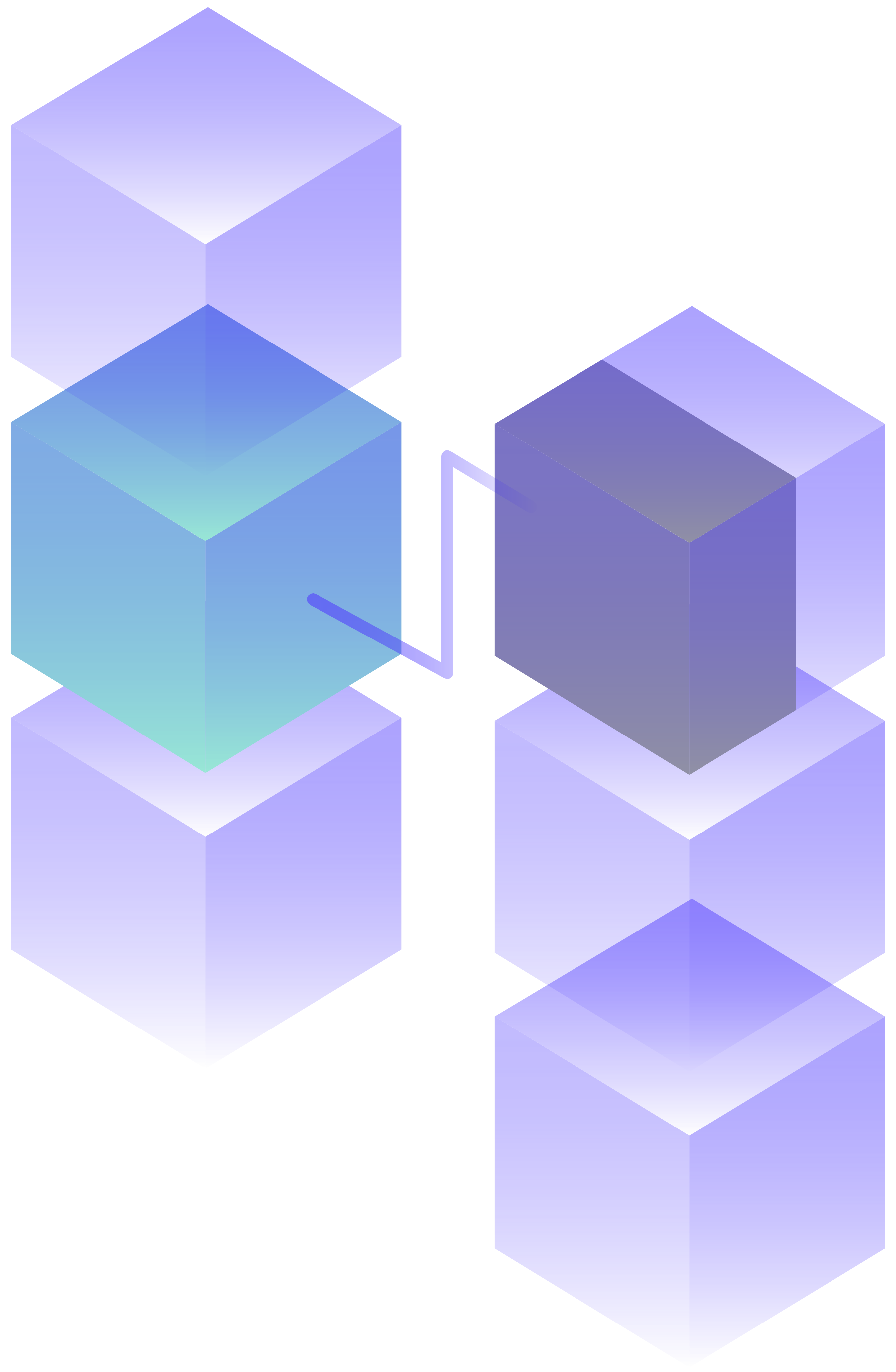
Hyperlinks can also facilitate the **dynamic inclusion** of content served by another application into the web interface of a self-contained system.



To further minimize coupling to other systems, synchronous remote calls inside the business logic should be **avoided**.

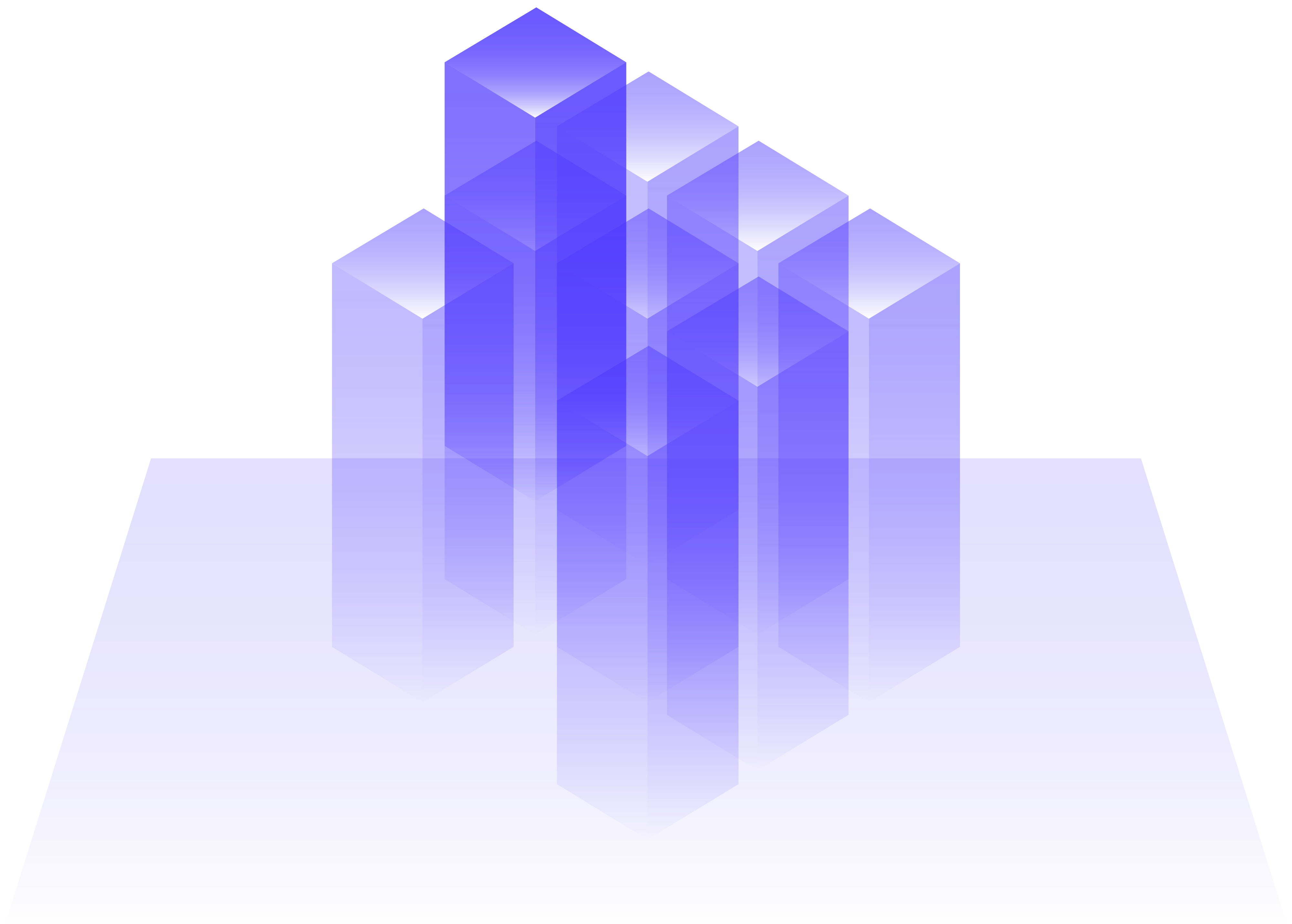


Instead, remote API calls should be handled **asynchronously** to minimize dependencies and prevent error cascades.

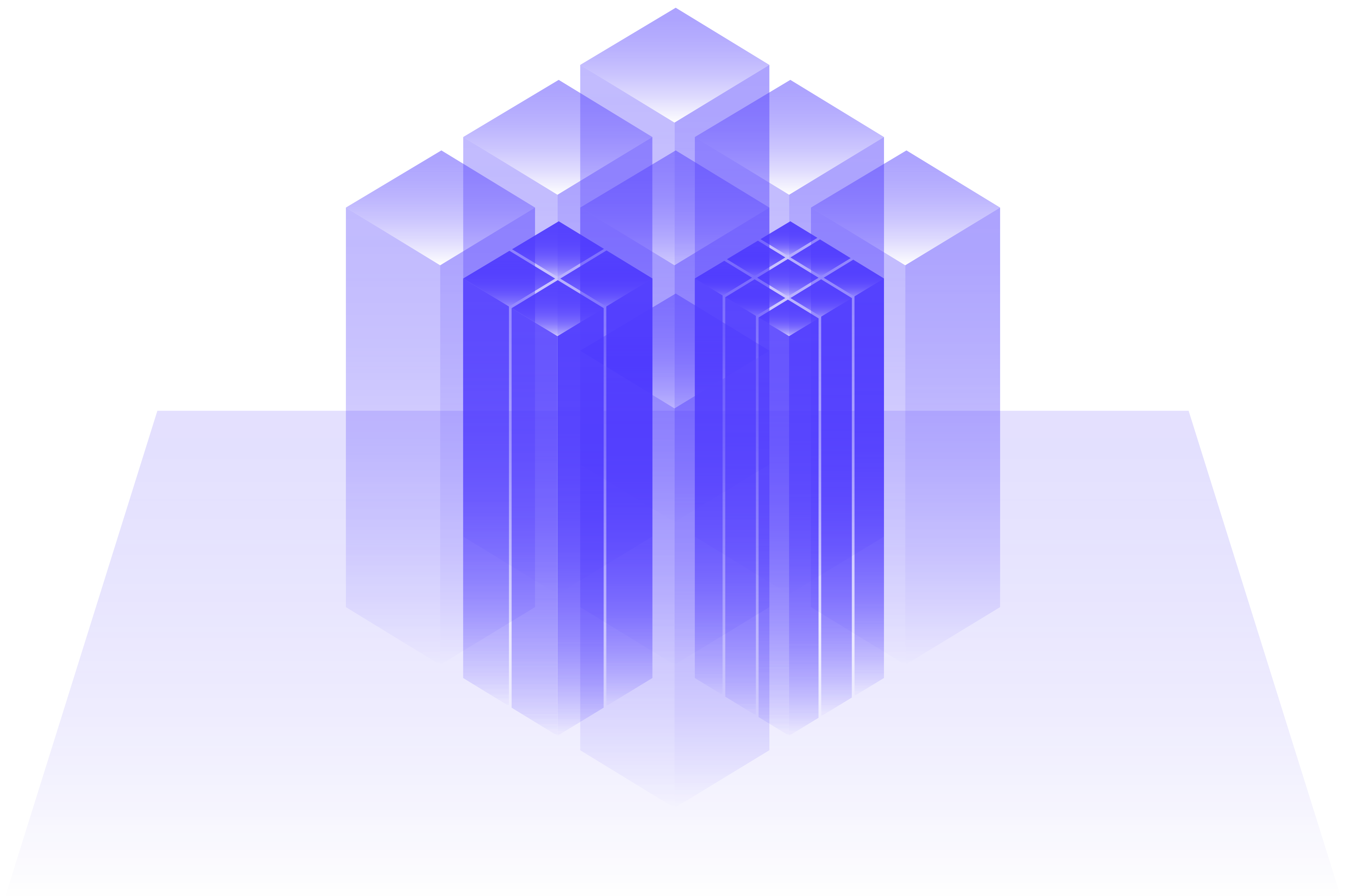


This implies that, depending on the desired rate of updates, the data model's consistency guarantees may be **relaxed**.

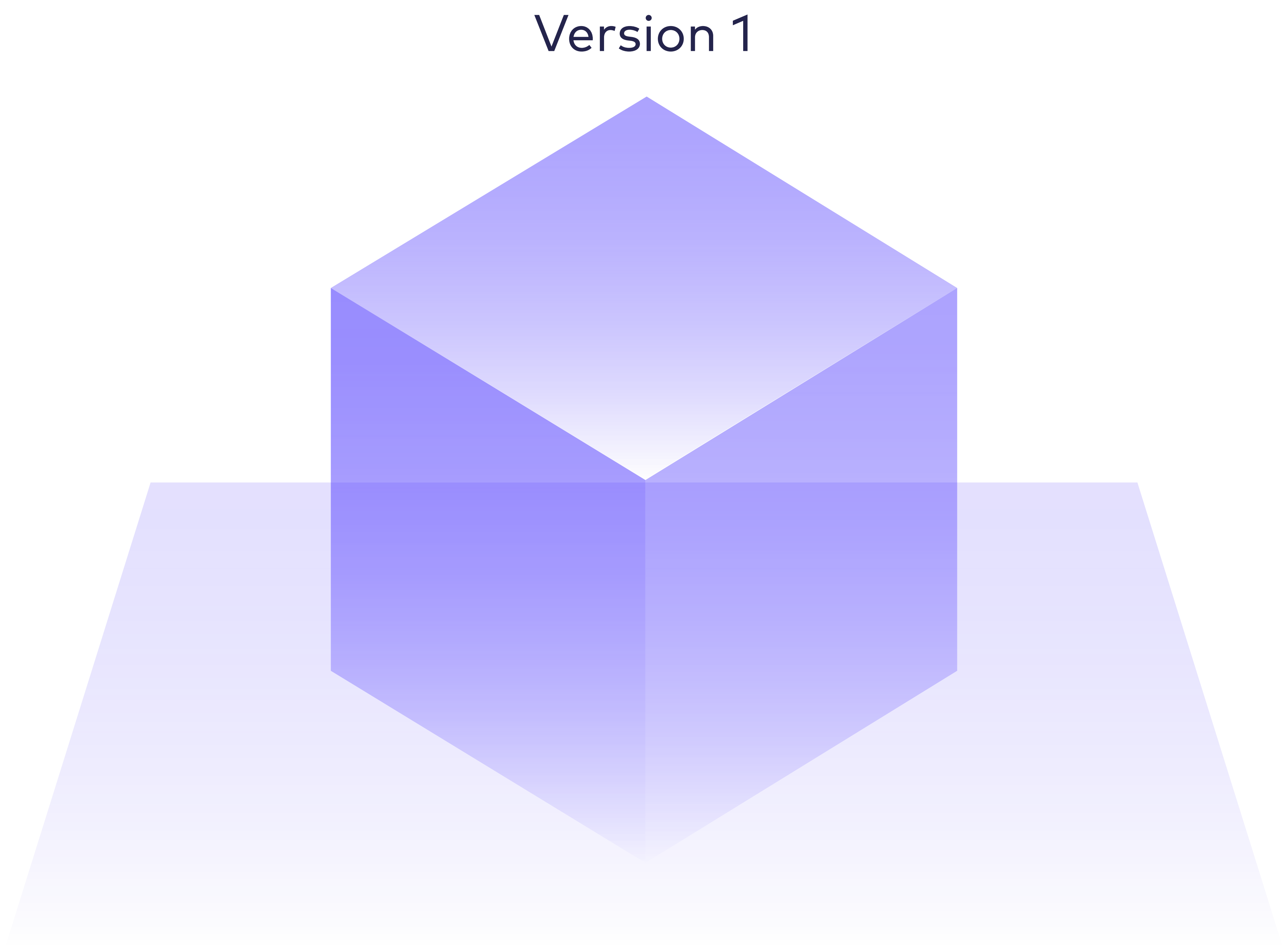
An integrated
system of systems
like this has many benefits.



Overall, **resilience** is improved through loosely coupled, replaceable systems.



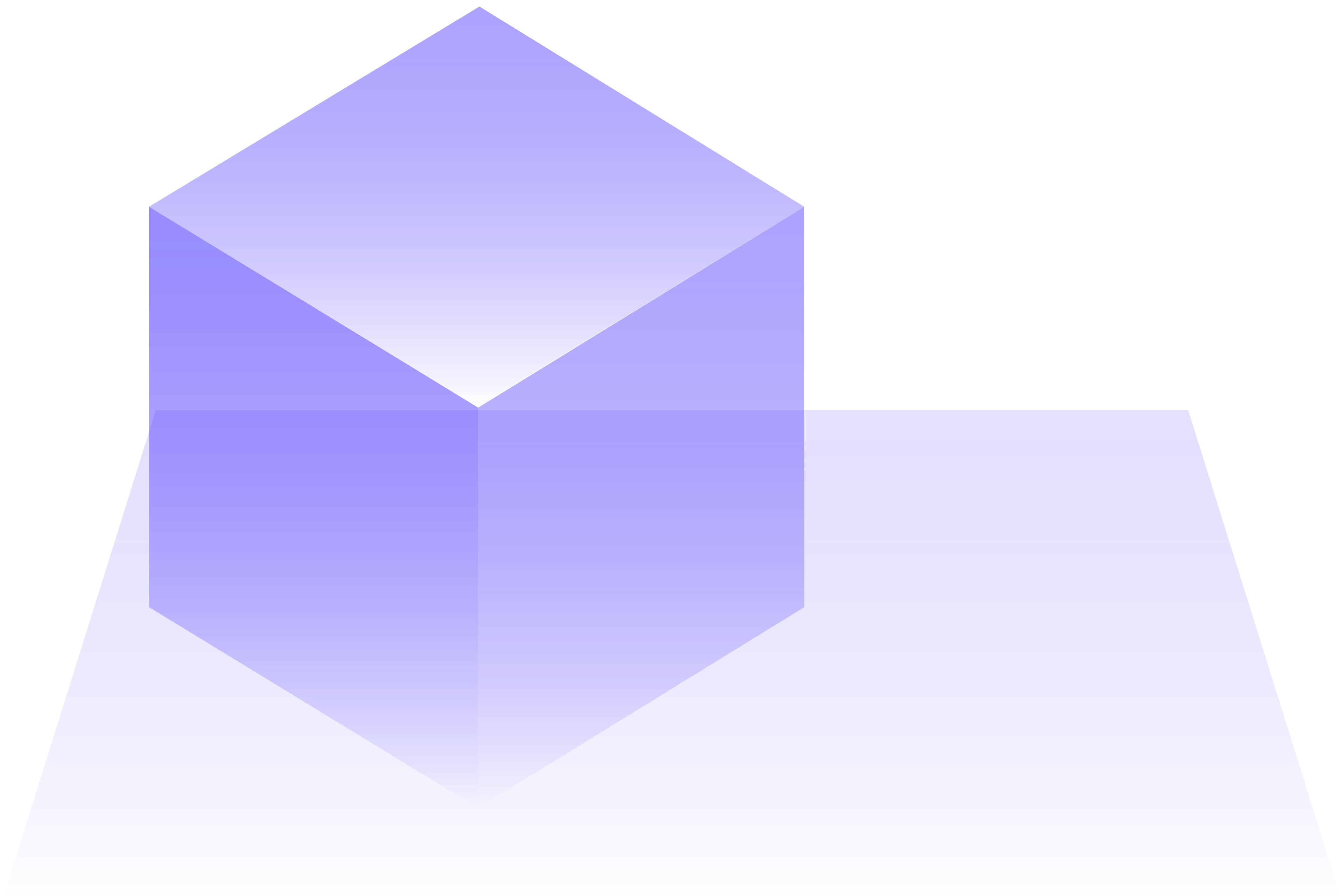
Some systems can be **individually scaled** to serve varying demands.



It's not necessary to carry out a risky **big bang release** to migrate an outdated, monolithic system into a system of systems.

Version 2

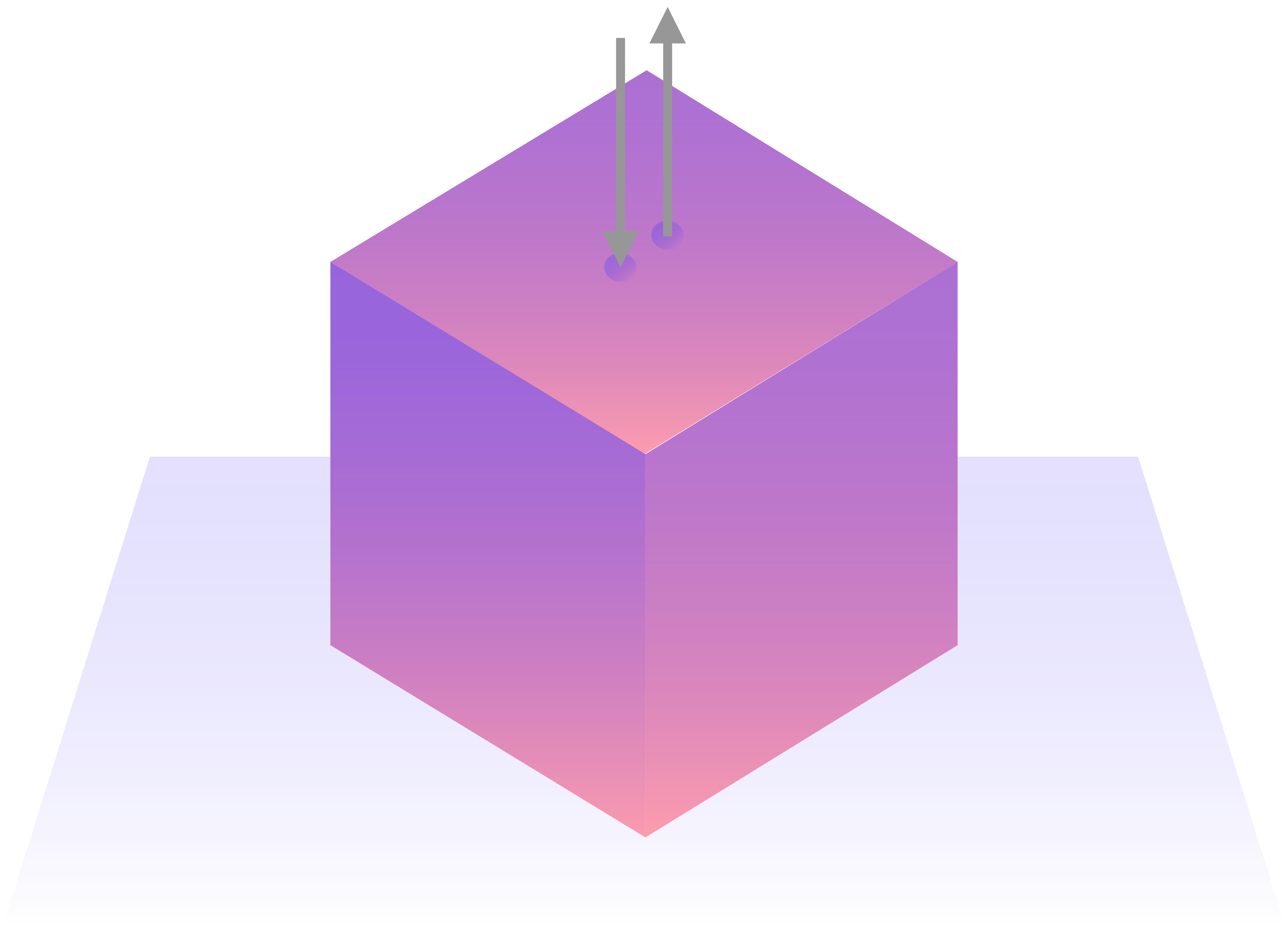
It's not necessary to carry out a risky **big bang release** to migrate an outdated, monolithic system into a system of systems.



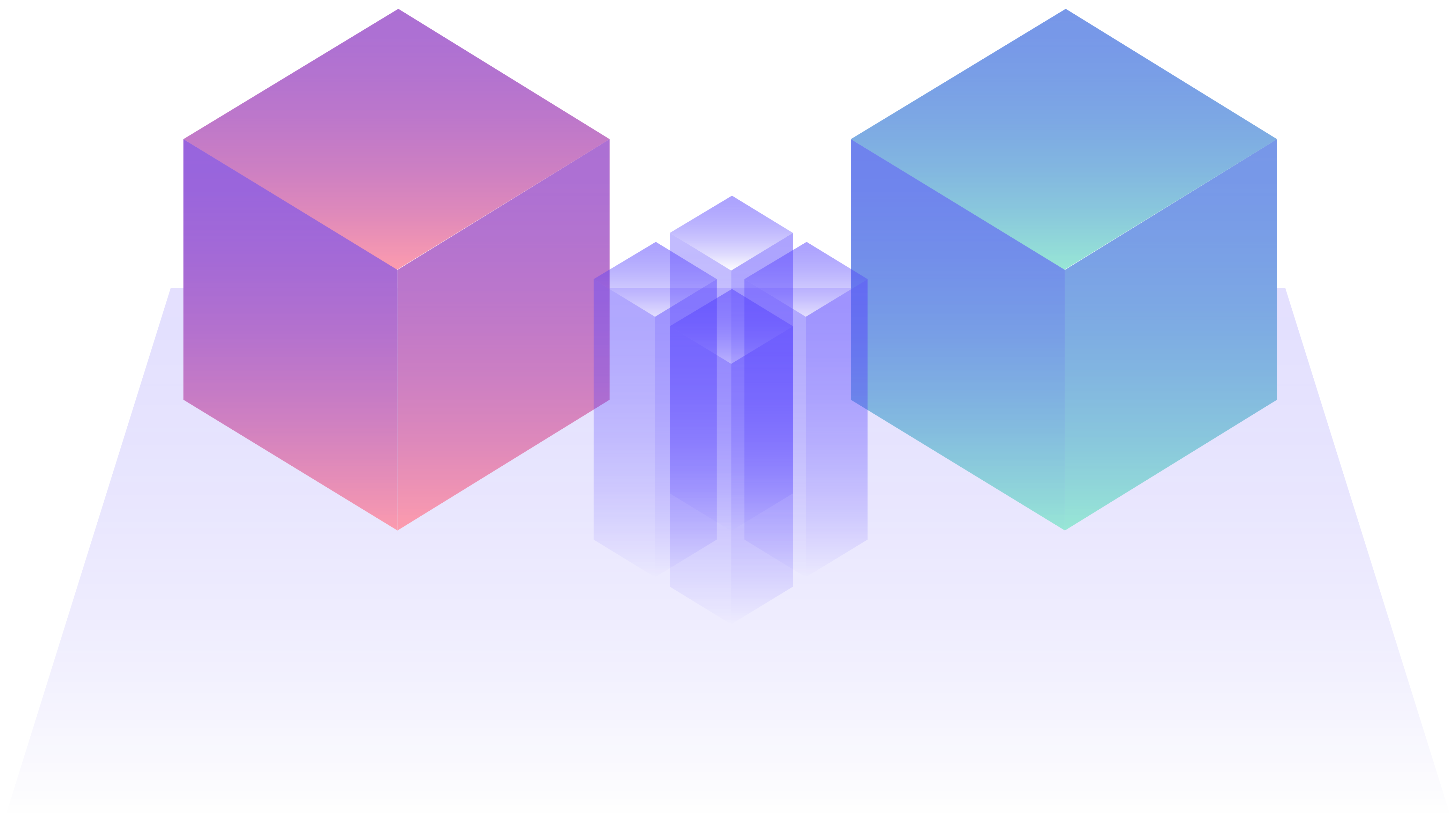
Instead, migration can occur in small, manageable steps that minimize the risk of failure and lead to an **evolutionary modernization** of large and complex systems.

Instead, migration can occur in small, manageable steps that minimize the risk of failure and lead to an **evolutionary modernization** of large and complex systems.

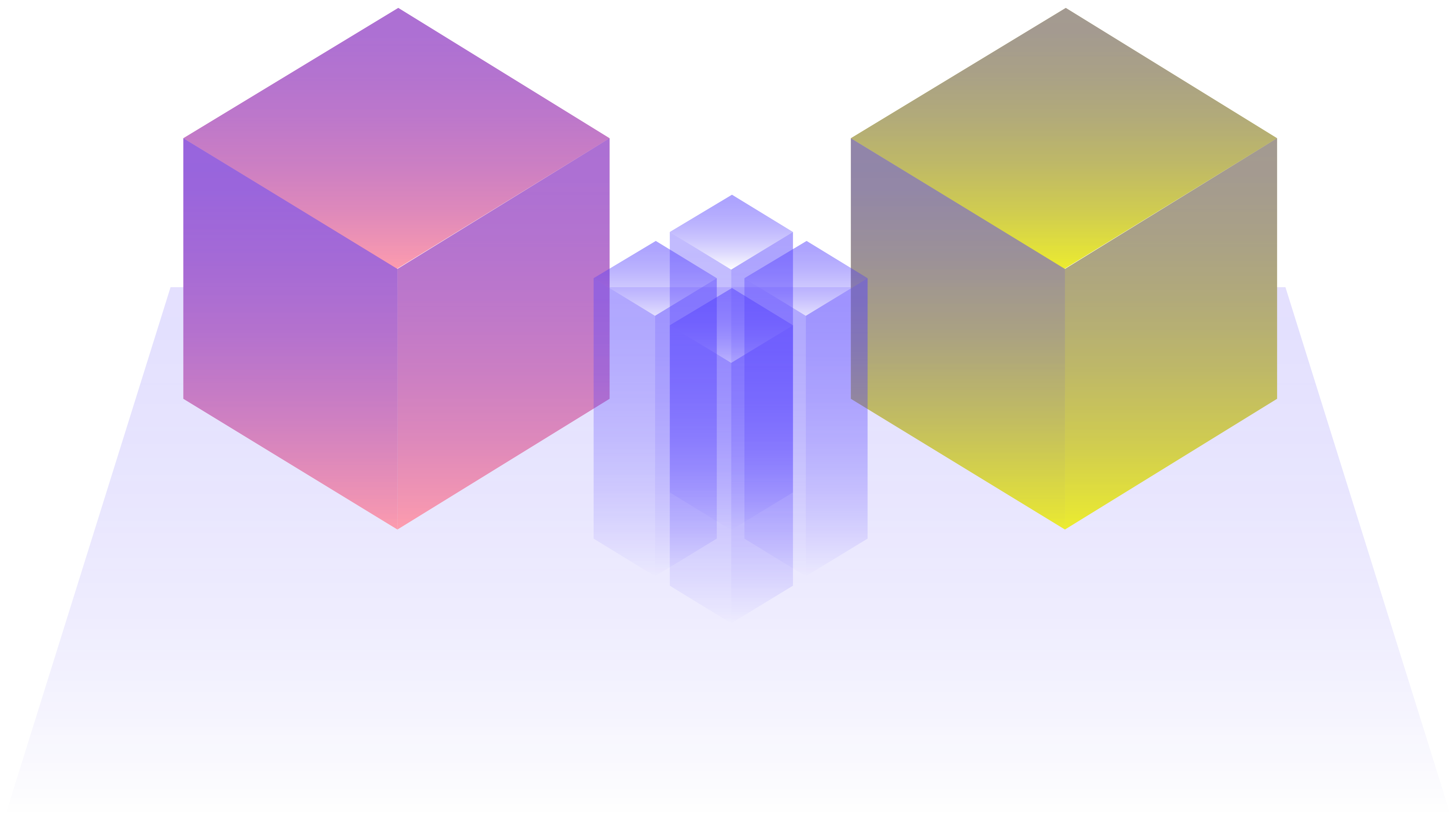
In reality a system of systems consists of individually developed software **and** standard products.



A product that fits well into a system of systems can be selected based on the following aspects: it should solve a **defined set of tasks** and provide the same **integration mechanisms** that a self-contained system offers.



This ensures that products can be **replaced safely** by other products once their lifetime has ended.



This ensures that products can be **replaced safely** by other products once their lifetime has ended.

If a product with such integration mechanisms cannot be found, it should at least be possible to extend that product with **uniform interfaces** that integrate smoothly with the rest of the system.

www.innoq.com

You can explore more in-depth information about self-contained systems, microservices, monoliths, REST, or ROCA at innoq.com/scs

Looking to modernize your IT landscape?
Or build something new?
We'd love to assist you. Get in touch!