Self-Contained Systems
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 consists of web technologies that are composed according to ROCA principles.
Besides a web interface a self-contained system can provide an *optional* API.
The business logic part only solves problems that arise in its core domain. This logic is only shared with other systems over a well defined interface.
The business logic can consist of **microservices** 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**, which means a database can be chosen to solve a domain specific problem rather than to fulfill a technical urge.
Inside of a self-contained system a bunch of **technical decisions** can be made independently from other systems, such as 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 over their web interfaces to minimize coupling to other systems.
Therefore simple **hyperlinks** can be used to navigate between systems.
A redirection can be used to ensure navigation works in both directions.
Hyperlinks can also support the **dynamic inclusion** of content that is 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 reduce dependencies and prevent error cascades.
This implies that – depending on the desired rate of updates – the data model’s consistency guarantees are 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 is not necessary to perform a risky **big bang release** to migrate an outdated, monolithic system into a system of systems.
It is not necessary to perform a risky **big bang** release to migrate an outdated, monolithic system into a system of systems.
Instead a migration can happen in small, manageable steps which minimize the risk of failure and lead to an *evolutionary modernization* of big and complex systems.
Instead a migration can happen in small, manageable steps which minimize the risk of failure and lead to an **evolutionary modernization** of big and complex systems.
In reality a system of systems consists of individually developed software and standard products.
A product that fits well in a system of systems can be chosen by the following aspects: It has to 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 can not be found, it should at least be possible to extend that product with uniform interfaces that integrate well with the rest of the system.
You can find more interesting content about self-contained systems, microservices, monoliths, REST or ROCA at https://www.innoq.com

If you have questions or feedback please do not hesitate to contact us info@innoq.com