# **Technical/Architectural Design Specifications**

## Architecture Requirements

IDM will be developed based on Microservices architecture that includes n-tier as an architecture of each service. Microservices architecture, being an industry-proven software architecture model is suitable to support enterprise-level client/server applications by resolving issues like scalability, security, fault tolerance, etc.

Microservices - also known as the microservice architecture structures an application as a collection of services that are:

* Highly maintainable and testable
* Loosely coupled
* Independently deployable
* Organized around business capabilities

Being developed based on the Microservices, Synergy IDM is a fully scalable web application deployed in the Kubernetes environment. Kubernetes is an open-source container-orchestration system for automating application deployment, scaling, and management. Kubernetes itself supports scalability and load balancing for micro-services.

In Microservices architecture, the solution will be designed to consist of multiple components that will be deployed as multiple services. Within each service n-tier architecture is used. Each service has its own service layer, business logic layer, and persistence layer. The services are developed and deployed independently of one another. Each service has its own database to be decoupled from other services. However, the services do not function independently, but will interact and communicate with one another through defined interfaces.



*n-tier architecture vs. Microservices architecture*

Being an application designed by Microservices architecture, the solution will have the following advantages over applications adhering to other types of technical architectures (e.g., monolithic, etc.) which also fully covers the benefits laid out in the TOR, including:



*Advantages of the Microservices Architecture*

**Secure:** Microservices architecture will provide better and finer security control as it is possible to enforce different level of security to different services if security requirement differs from service to service.

**Easy to manage:** Microservices architecture will enable to manage each service separately, without affecting the other tiers. Since each service will have its own presentation layer, service layer, business logic layer, and persistence layer, it will allow to easily manage the system.

**Scalable:** Microservices Architecture is highly scalable both vertically and horizontally due to its capability of multiple service deployment and service decoupling.



**Flexible:** Implementing Microservices architecture for the solution provides a more effective solution for the system, its users and other parties involved as business processes and data sources tend to change over time. It will be possible to change or upgrade one service without affecting other services. This means that pension department can add more components without having to rewrite the entire application or redesigning the whole software, thus making it easier to scale or maintain.

**More efficient development:** It will be friendly and efficient for development as it will be easy to co-work on the different services. It will also allow to modify in support of changing business rules and there is less risk modifying the code that implements any given business rule.

**Easy to add new features:** Microservices architecture will allow to add new feature/module without affecting other services.

**Easy to reuse:** It will offer the highest potential for service reuse and sharing.

Moreover, application partitioning will be used for the application to flexibly distribute application logic for optimal performance. The partitioning of the application components will be used to achieve the following major benefits:

* It reduces the turnaround time for the data result.
* It decreases the amount of network traffic necessary to transfer the data to the client.
* It supports multiple, diverse hardware/software configurations.
* It enables object and component reuse as services can be shared within and among applications.
* It ensures separation of business rules from presentation and data.

Besides, application partitioning overcomes the limitations of previous renditions of client/server architectures, such as overloaded clients, low reliability, reduced performance and network bandwidth, inflexibility, and high maintenance. In the system, the processing load will be distributed across multiple nodes. This will allow maximizing the benefits of the Microservices model.

Because microservices approach will be used to develop the E-Platform, it will be important to note that microservices will interact with each other with strict and fixed protocols. This approach will result in the application being easier to maintain and to enhance because coupling is reduced in services.

Currently, Synergy IDM has over 40 services. Each Microservice has its own service layer, business logic layer, and persistence layer.



*Microservices Architecture*

**Presentation Layer**

The presentation layer (user interface) provides the user means to interact with the application through intuitive screens that employ techniques to make the application easier to learn and use. The graphical user interface (GUI) will target the most popular desktop environments and support the business requirements of the system. The presentation layer will provide means for the user to enter new data, view/edit existing data, run pre-defined reports, and perform administrative tasks.

The technologies that will be used to build the presentation layer include but are not limited to:

* Angular JS
* TypeScript
* MDL
* Sass
* Kendo UI
* jQuery
* SVG

**Service Layer**

The Microservices used to develop IDM systems interact with each other through strict and fixed protocols. This approach results in the system being easier to maintain and to enhance as coupling is reduced in services.

More specifically, the web services developed within the scope of the IDM projects are implemented based on the Representational State Transfer (REST), a decoupled architectural style and lightweight approach to communications emphasizing that interactions between client and service are enhanced by having a limited number of operations. These features make REST an ideal choice for application program interfaces (APIs) that are used to build web services for the IDM projects.

**Business Layer**

The IDM-based solutions are designed to have a separate business layer for each microservice that encapsulates and handles all the business domains and logic. It is used for processing complex business rules, transforming data, applying policies, and for validation. It serves as a means for decoupling business logic from presentation and data access code and for simplifying the testing of business logic.

Having a separate layer for business rules and their processing improves the maintainability of the system. The business layer in IDM is used to store the business rules on how to retrieve the data from the databases and then how to pass the retrieved data to the server that takes this information to display on the presentation layer.

The business layer is used to ensure cross-layer communication through the messaging middleware (a messaging method of communication between software components and different applications), REST service, and APIs. The technologies that are used to build the business layer and interaction between presentation layer and business layer include, but are not limited to:

* Spring MVC
* Elasticsearch
* SL4j
* Log4j2
* Rest services
* Jackson
* JSON

**Persistence Layer**

The persistence layer encapsulates the behavior needed to make data persistent, that is to say, write, and delete data to and from a permanent storage, e.g. a database. Each system component stores the data in its own database. The database manages data storage, structure, access, and security. Data can be organized into related tables so that relationships between and among the data can be established. The database is implemented on a separate layer from the presentation and business objects.

Data synchronization is used across the databases that comprise the persistence layer for keeping them up to date with each other's data changes. This helps to avoid data discrepancy and ensures data consistency in different services. This way, the data stored in the data entry tables are synchronized with the Elastic search service. In the same way, the data are synchronized with the IDM Analytical Platform. The technologies that are used to build the persistence layer include, but are not limited to:

* EH Cache
* Spring JDBC
* C3P0
* HikariCP

Currently, Synergy IDM has over 40 services. Each microservice has its own presentation layer, business logic, and persistence layer. A brief discussion of different microservices to be included in the e-Platform and the interactions between these microservices is presented in the following diagrams.



*Technology Services Interconnections*

## Deployment Requirements

The microservice architecture enables the continuous delivery/deployment of complex enterprise-level client/server applications. It also enables an organization to evolve its technology stack and resolves issues like scalability, security, fault tolerance, etc.

Services will be developed and deployed independently of one another. Each service will have its own database in order to be decoupled from other services. However, the services will not function independently, but will interact and communicate with one another through defined interfaces.

Being developed based on the Microservices, Synergy IDM is a fully scalable web application deployed in the Kubernetes environment. Kubernetes is an open-source container-orchestration system for automating application deployment, scaling, and management. Kubernetes itself supports scalability and load balancing for micro-services.