Project Proposal

DataAPI

1. Introduction

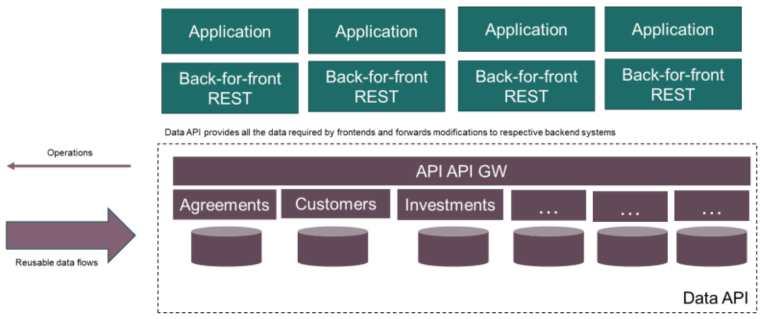
Mandatum builds new services based on modern microservice architecture. Large-scale adoption of this architecture has created second-order challenges in managing those services.

One of those challenges is that many applications need to connect to 20+ REST services to show the data that they need. This makes it difficult to make modern web and mobile applications robust and highly performant.

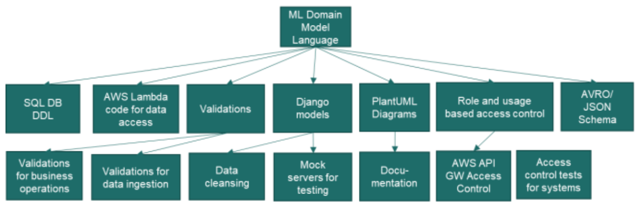
Also, business applications often need to offer a harmonized view of similar data across multiple backend systems. Typical data integration pattern for avoiding such point-to-point integrations is to turn this around, and make the backend systems to offer a view to data that is harmonized to a canonical data format.

The platform implemented by this project allows data to be accessed by applications through unified API, using such harmonized, versioned canonical data format. Building on in-memory databases and serverless technologies, the API can be highly performant and scalable.

When API and documentation provided by platform is generated from formal business domain specifications, we can ensure that the business domain models evolve in a managed way. During this project, we will create a Domain-Specific Language (DSL) to define such specifications.



For example, we may use this language to define a business domain model for context “Customers”. Within this context, we might have entities like “Customer”, “Address”, “Organization” and so on. These entities have attributes, for example “Customer” might have “first_name” attribute. They also might have relationships between them, such as “Customer belongs to one or more organizations”.


ML Domain Model Language
SQL DB DDL
AWS Lambda code for data access
Validations
Django models
PlantUML Diagrams
Role and usage based access control
AVRO/ JSON Schema
Validations for business operations
Validations for data ingestion
Data cleansing
Mock servers for testing
Docu-mentation
AWS API GW Access Control
Access control tests for systems

- Formal, versioned domain model definition (under version control)
- Integration data format = application API data format

➔ Automatically generate most of the Data API code and documentation, including validations and access control

We can see that the specification language allows us to define something that resembles POJO objects in Java or Python Django models. Although could define what we need by using program language annotations, that would limit the readability of the specification. It would also make adding some features beyond the annotations difficult. (These features are beyond the scope of this project.) Therefore, we have decided to develop our own Domain-Specific Language (DSL) for this purpose.

DataAPI is part of larger integration architecture reorganization at Mandatum Life. Still, it may be used as standalone platform by other organizations if it is published as open source.

The integration architecture follows the Kappa architecture pattern, and DataAPI provides the “materialized view” part of this pattern. The project members will become familiar with the latest research findings on data architectures, schema evolution and probably use the latest serverless technologies on cloud platforms.

2. Project goals

The project should deliver a data-oriented, documented, scalable, highly available and performant platform that provides a REST API to access to business data stored on the platform. Data comes into the platform through asynchronous messaging data streams in canonical data format.

The API and incoming data importers should be dynamically generated based on business domain model specification. The specification should be defined in a Domain-Specific Language (DSL), which does not yet exist beyond some prototype, but will be defined as a part of this project.

The specification (and the API) must also support industry-specific requirements for controlling the access to data.

The platform must also maintain the integrity of the data.

3. Technologies and Methodologies

Programming language for the platform is either Python or some JVM-based language, depending on team interest and skills.

Other technologies and methodologies will depend on the final architecture, and we expect team to participate in that discussion and choice based on skills by team members. Here are some strong candidates:

- Apache Kafka / Confluent platform
- AWS Lambda / Serverless framework
- AWS RDS or AWS Aurora or in-memory / NewSQL databases
- Domain-Specific Languages (DSL)
- Domain-Driven Design (DDD)
- Event Sourcing / Kappa Architecture
- RESTful APIs

We believe the best way to learn is to do by yourself. However, if you get badly stuck, we will probably be able to provide some technical assistance if the same technologies are chosen which we use internally, such as those mentioned above.

4. Requirements for the students

There are no hard requirements for students. Beyond programming skills, good attitude towards getting things done and willingness towards learning new things will be necessary.

Interest in Domain-Specific Languages (DSL) and programming languages with metaprogramming features, such as Clojure or other LISP dialects will probably be very useful in this project.

The difficulty level of the project is demanding, both in terms of technologies and domain.

While the technologies themselves are quite straight-forward, due to non-functional requirements (performance, scalability, availability), these technologies need to be understood beyond simple basic usage. However, it may be possible to learn these sufficiently during the project if you are really motivated to learn.

The complexity in the domain is mainly due to abstract thinking required when developing Domain-Specific Languages. There is no need to understand the business domain; sufficient examples will be provided by the Client.

5. Legal Issues

Intellectual Property Rights (IPR):

The client gets all IPRs to the results. If some of the project team members wish, the code may additionally be published under MIT compatible license after the project. The team members will be mentioned on this project, but the company mentions will be worked out with our legal department.

Confidentiality:

The client will share some confidential information with the students. We put this clause here to satisfy our legal team and security policies. This is needed so we can more freely share some internal architecture diagrams and other such documents.

If we provide the students with some computing resources, such as an AWS account, access to these resources may require signing an additional contract directly related to fair use of those resources (no bitcoin mining, no criminal activities, etc.).

6. Client

About the client

Mandatum Life is one of Finland's most respected and solvent financial services providers. We offer private, corporate and institutional customers comprehensive services for building wealth and preparing for the future. Mandatum Life’s services include unit-linked insurance and wealth management, saving and investing services, personal insurance, pension and reward solutions for companies, as well as related consultation services.

With more than 500 employees, Mandatum Life is a workplace that strives to create an inspiring work community and to work as one. Mandatum Life has been recognised as one of Finland’s best places to work for the ninth year in a row in the Great Place to Work Finland survey.

Client representatives

Product Owner will be Mikko Ahonen, ext-mikko.ahonen@mandatumlife.fi, +358 50 343 6393.

Mikko works as System Architect and CTO at Mandatum Life. Mikko 25 years of IT industry experience in various positions. He has worked on few proof-of-concepts related to the project.

Location

Due to the current situation, we will not be able provide office space at our headquarters. If the team wishes to work at physical location, we will organize a suitable physical work space, either near our offices or on Aalto campus.

Sprint demos, unless done remotely, will be held at Mandatum Life HQ at Bulevardi, Helsinki.

Computing Resources

Client will provide the team with an AWS account or other similar resources if needed for the project.

Preselected Student Team Members

None.

7. Additional information

Documentation and implementation language is English. Other communication may be in Finnish or English, depending on the team.

For version control, we will use git. Otherwise, the team may choose their preferred collaboration tools. At Mandatum Life, we use Teams, Jira and GitLab.