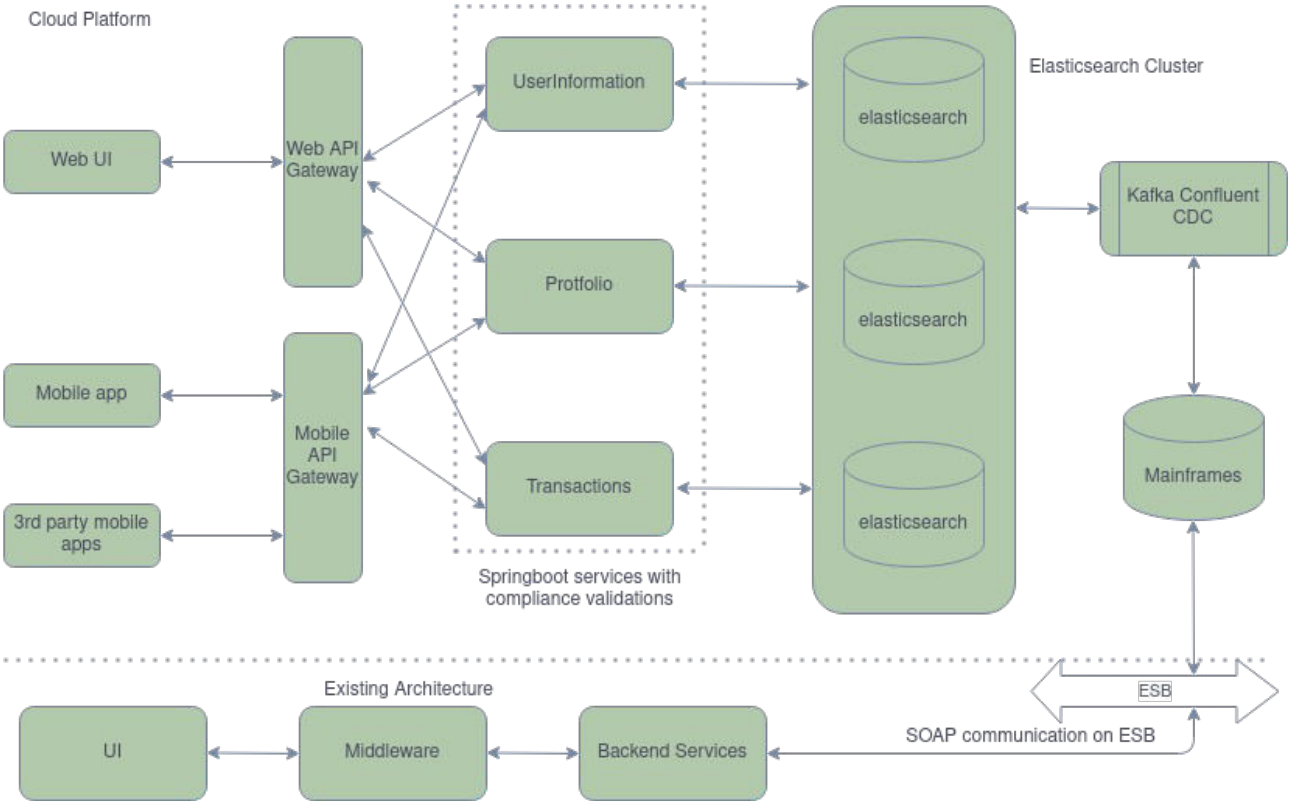
Mobile App for Banking

Sotfware Architecture



Overview

In the above image existing architecture, backend services communicate through SOAP which can add to load on existing system, not scalable and not reusable either. Due to this limitation we have to move away from the current existing architecture.

Springboot restful resources expose API to GUI and third party applications. Consideration of two different API gateway adds an extra layer and complexity to system especially wrt maintenance. However the drawback is over come by the flexibility, agility and scalability of the system.

Changes in mobile based applications will not cause a downtime of desktop based applications also any changes in a particular gateway will not affect the overall architecture. This platform is deployed on a cloud provider and adds to high availability of the platform. When we follow blue green deployment we can acheive zero downtime as well. Therefore the existing services which are already present can be reused for serving different IoT devices at a later point in time with an addition of new API gateway while the platform itself becomes highly scalable and available. With the new modular services we get an oppurtunity to add new compliance addition as well like the GDPR, PCI, PCII, etc in finance domain.

Delivery

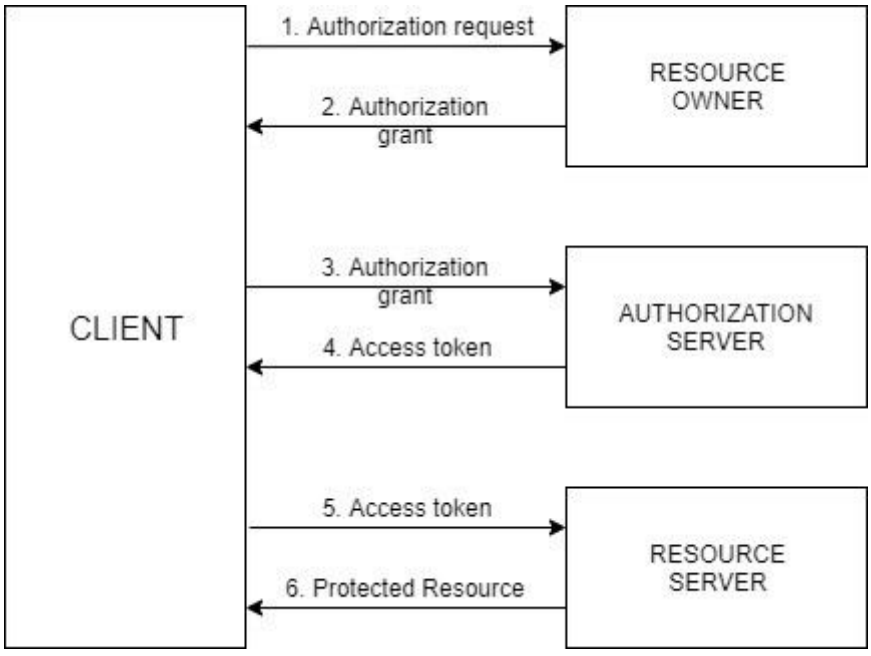
1. List of APIs and rationale behind the design

We are going to follow best practices and standards for creating resource URLs. Since its a new mobile app we are going to provide very simple CRUD operations to perform and most of them will be to view status of the account. For supporting multiple tenants we need to provide tenantId statically accross all APIs.

HTTP Method	URL	Rationale
GET	*/accounts/{id}*	Details of the particular account like branch details, account details
	/accounts/{id}/transactions?startDate=&endDate=	Transaction done for a particular period mostly limited to a constant period
	/user/{id}/accounts	All accounts of a user in the bank
	/user/{id}	User profile information like personal information and contact details
	/accounts/{id}/tax	Taxes on a particular account
POST	*/accounts/{id}/services/{type}*	Have a requestbody, request for a cheque book, FD, RD, etc.,
PUT	*/accounts/{id}/transfer*	Have a requestbody with details and type for a particular transaction and credit or debit, amount, etc.,
DELETE	*/accounts/{id}/services/{type}*	Cancellation request for a cheque book, FD, RD, etc.,

kaveats : Transaction status is not available yet also we cannot delete accounts as it is required for audit purposes maybe. We start with first things first so only basic set of operations provided.

2. Implementation of one of the APIs considering non-functional & security requirements



The above sample has got all the main pointers that are required to implement an Authentication and Authorization system. I have used the follow technical stack.

1. J2EE
2. Springboot
3. Spring-security
4. Maven
5. Jersey JAX-WS

3. Logical design and Technical stack merits and demerits

Demerits :

1. Will not help in complete migration immediately. Its during the course of time we have to get it done.
2. The platform conversation is not highly secure, have to implement security upon that.
3. Have to implement everything from scratch so going to consume time with the new implementation and have to deal with addition application complexity.
4. Deployment complexity is added.
5. A small lag in time as its a distributed and eventually consistent system. So we may not be able to see the changes immediately.

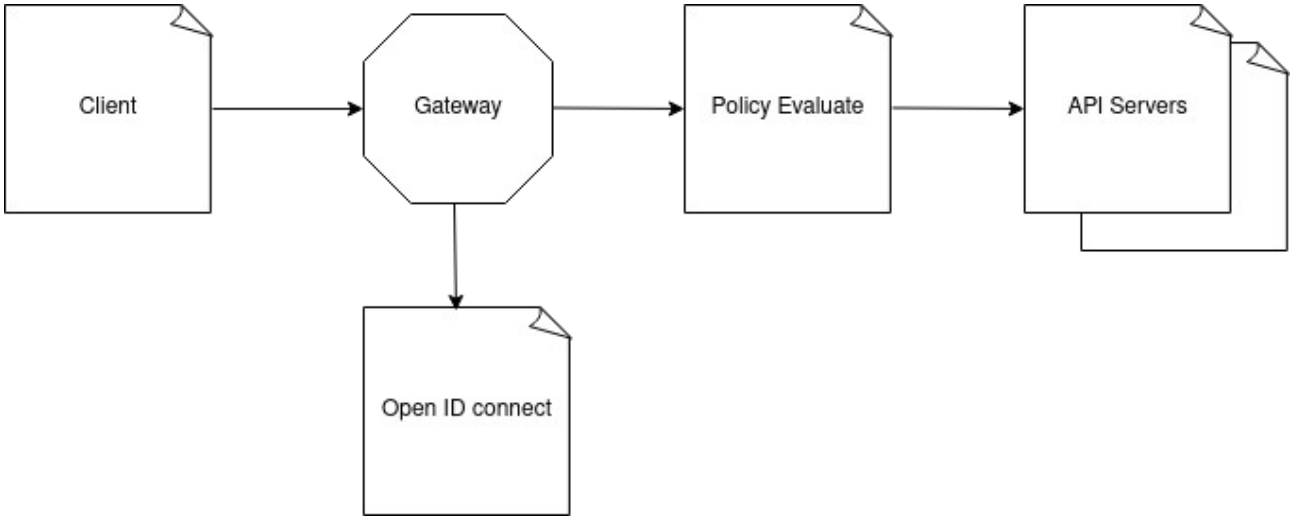
Merits :

1. Flexible, scalable and reliable microservices platform, helps in parallel development.
2. Helps in debugging and bugs finding easily as each service is relatively small.
3. Most of the merit i have mentioned in the overview part as well.

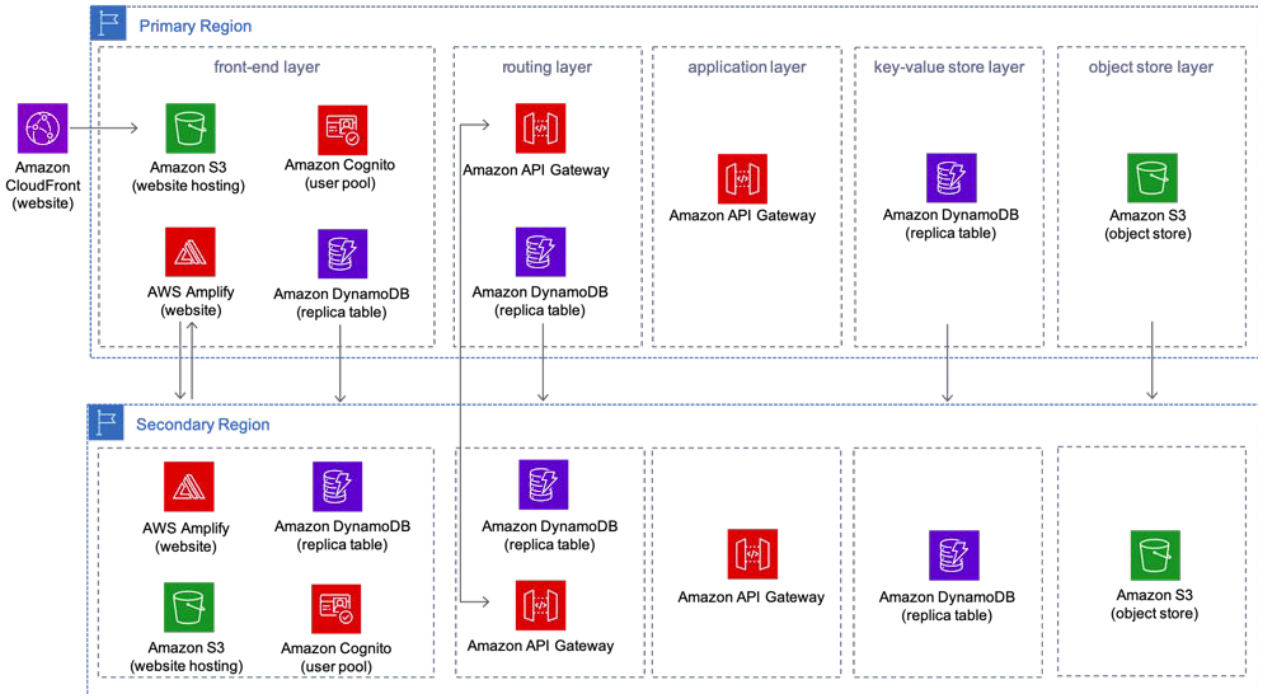
Below are the other details requested for :

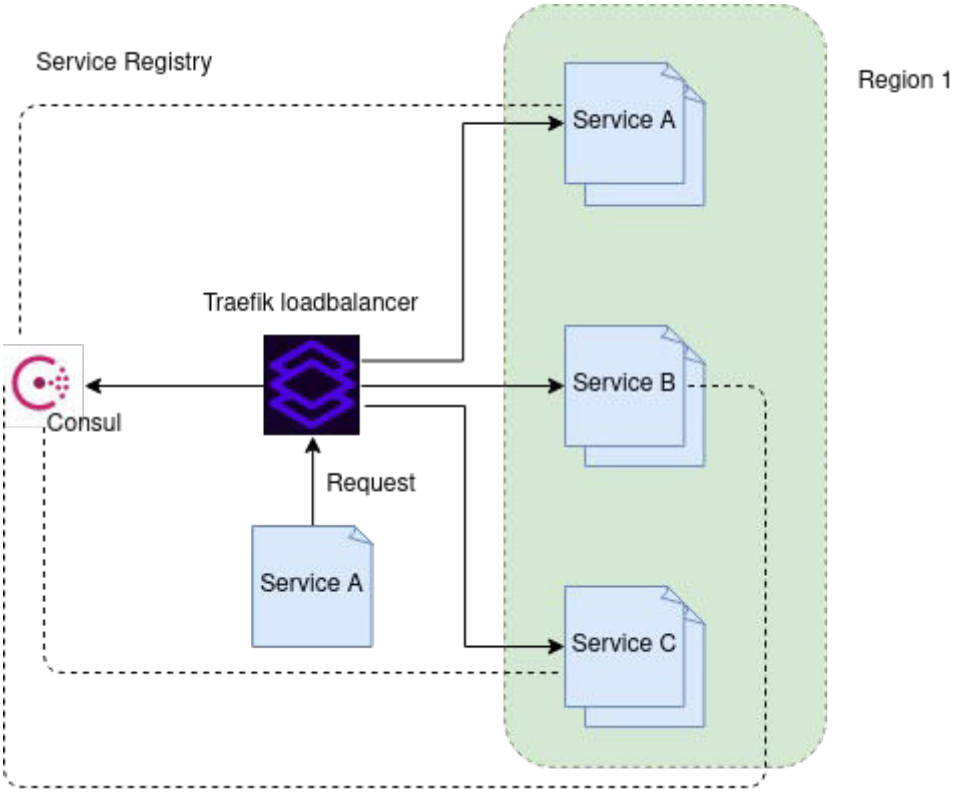
- Logical architecture to move from SOA to API’s
- Support for multiple country
- Deployment architecture (detailed)
- Performance
- GDPR implementation
- Security(federated)

Logical Architecture



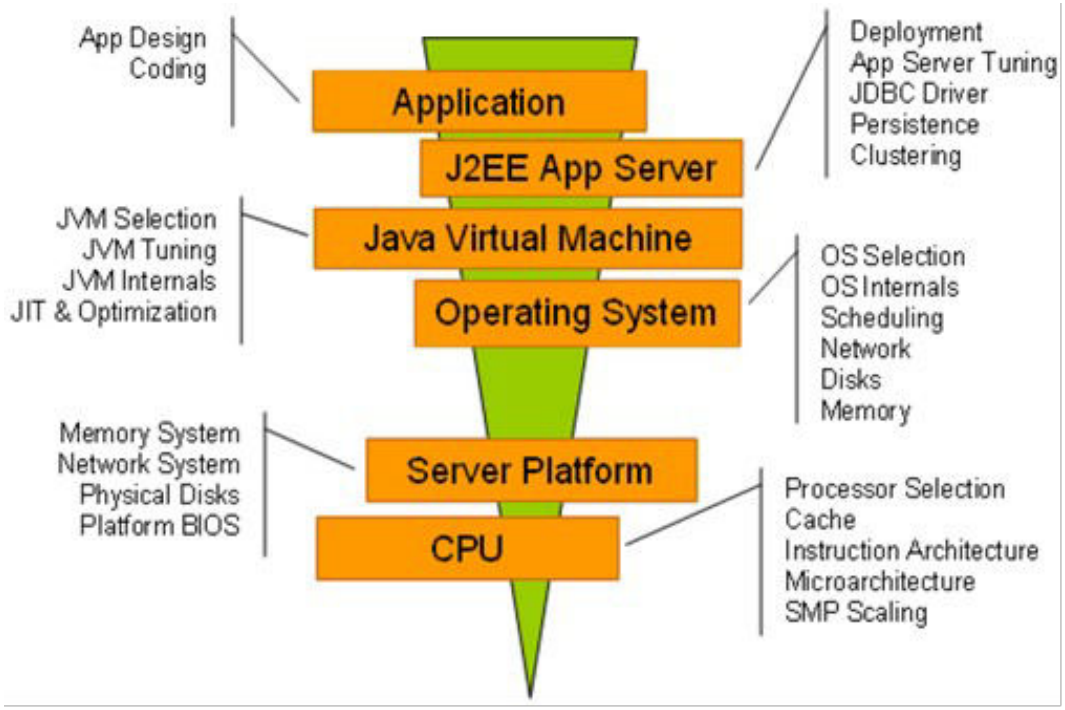
Support for Multiple Country and detailed deployment architecture



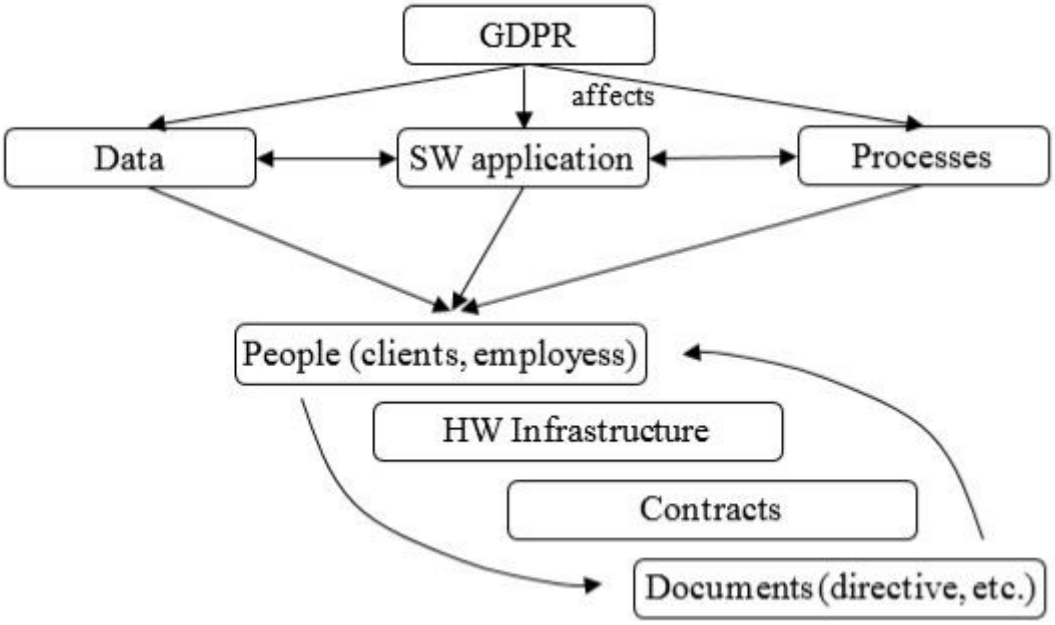


Performance

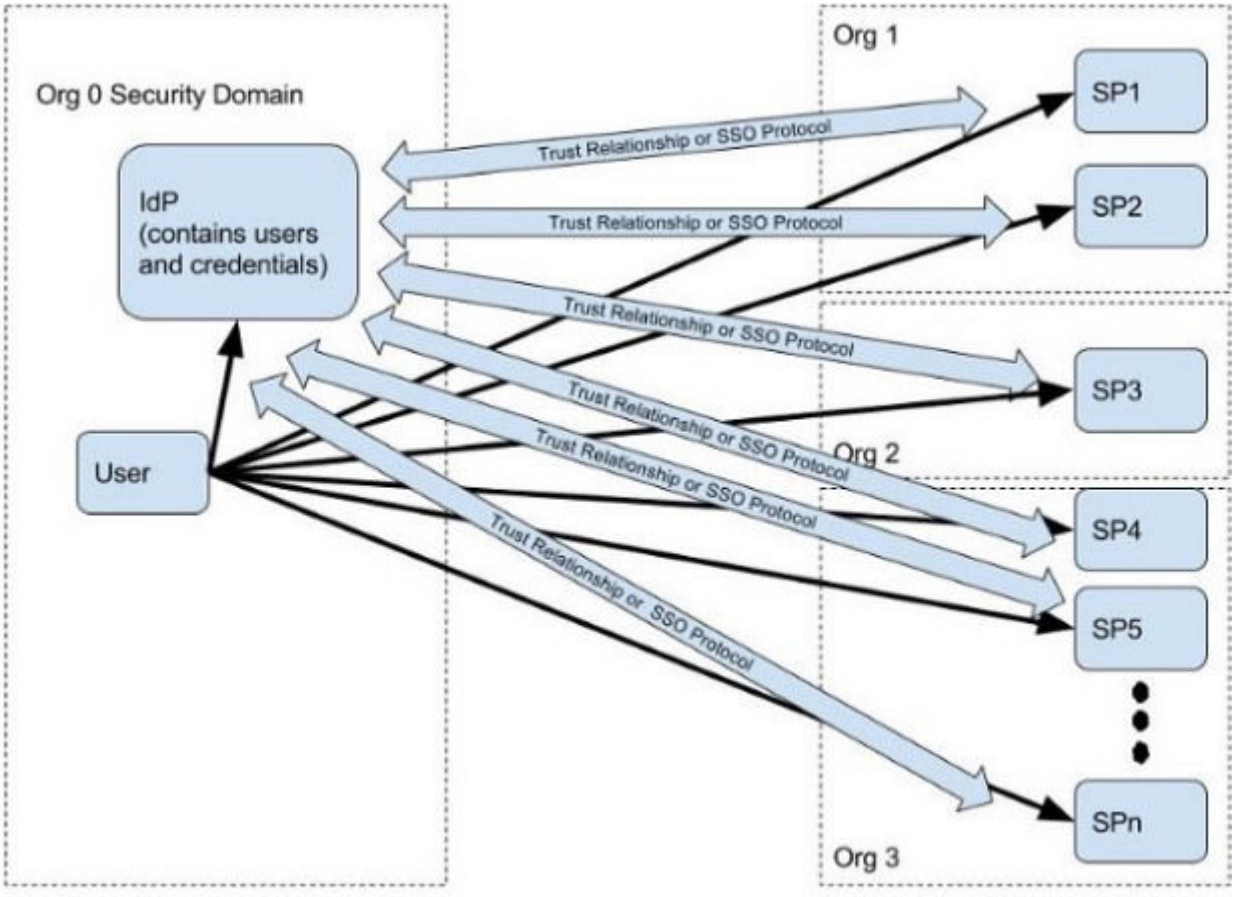
Various layers of performance tuning that can be provided



GDPR Implementation



Security Federated



N SPs across multiple organizations trusting a single third-party IdP