My justifications on my design point of view are:

I looked to the components of the current system and its functionality, I found that fetching person’s data is the most crucial job inside the system and it is a repetitive task that called frequently, So I thought delegating this task to a fast separate microservice is a good idea.

Since gRPC messaging provides the greatest performance over other messaging types I choose it to implement a separate gRPC microservice for fetching person’s data only naming it by "grpc-persons-api".

Other services will relies on it for fetching these data when needed like the UdaConnect router service itself and the "Connections Service" also.

On the other side, a "Kafka" messaging services is a very good candidate to carry out the job of saving person 's data since it is a crucial data in the system and needs to be reliable and persisted efficiently, so I create a one named "kafka-persons-api" to carry out this job.

However since there is no frontend implementation for posting or altering employee's records this service will be invoked when the endpoint of the UdaConnect post persons endpoint.

So at the end we have:

1- UdaConnect api server which the frontend connects directly and it acts as router gateway for the client requests to redirect the requests to the other proper microservices of the system.

2-A gRPC microservice that is called a "grpc-persons-api".

3-A Kafak microservice that is called a "kafka-persons-api".

4-A Rest microservice that is called a "http-connections-api".

Each of them will be hosted in a separate worker node and act as microservice.