The Service Service nodes are the ones that do the work a client is interested in. They receive tasks, process them and send back responses. Processing requests is usually costly so we imply the help of gateways and caches. Each service has a database. For the services, you'll need to implement the following features:

- ☐ Long-running saga transactions;
- ☐ Database redundancy/replication + failover.
- ☐ Service instances connect to different DB replicas;
- ☐ ELK stack or (Prometheus + Grafana for logging).

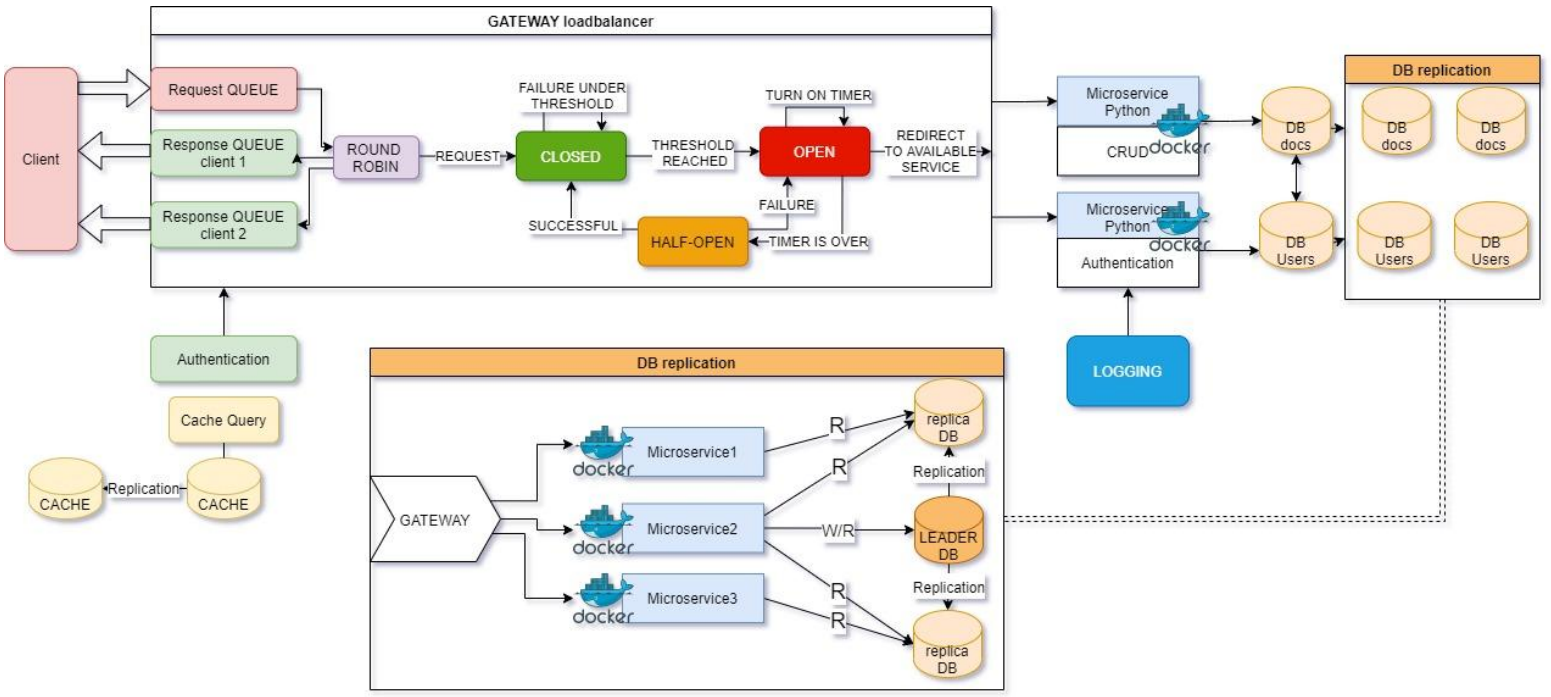
The Gateway The gateway is the node that receives and forwards user tasks to the service nodes. Based on some logic, the gateway caches responses and balances the load of service nodes. Finally, it has a service registry and chooses from registered services when load balancing. For the gateway service, you'll need to implement the following features:

- ☐ Service high availability (if a request to a service fails or the service is otherwise unavailable, route the request to a different one);
- ☐ Trip circuit breaker if multiple re-routes happen;
- ☐ ELK stack or (Prometheus + Grafana for logging);
- ★ Cache high availability;
- ★ Microservice-based 2 phase commits including distributed transactions.

The Cache A cache allows your system to temporary store responses given by your services and serve them without bothering the service nodes. Using caches makes your system more responsive. Usually caches use in-memory storage. For the cache service, you'll need to implement the following features:

- ★ Cache replication;
- ★ Distributed cache using consistency hashing.

This lab's main focus will be on distributed databases. In green are highlighted the planned features for implementation. That's why implementing ELK or Prometheus+Grafana for logging will be a good idea in order to see the changes, ups and downs. Also an implementation of database replication and services that connect to different DB replicas will be good to have, it will take off the load from one database and share it with multiple instances, so we have improvements in speed and quality of our system. For the gateway a service with high availability will be a great solution in case a service fails and the request should be routed to another available without the client knowing this happened, this way following the principle of transparency. Cache high availability and Cache replication will also be implemented for system improvements. Everything will be more clear during the lectures and studying the theme in depth. Below is a representation of the system.


GATEWAY loadbalancer
Client
Request QUEUE
Response QUEUE client 1
Response QUEUE client 2
ROUND ROBIN
REQUEST
FAILURE UNDER THRESHOLD
CLOSED
THRESHOLD REACHED
TURN ON TIMER
OPEN
REDIRECT TO AVAILABLE SERVICE
SUCCESSFUL
HALF-OPEN
FAILURE
TIMER IS OVER
Microservice Python
CRUD
Microservice Python
Authentication
DB docs
DB Users
DB replication
Authentication
Cache Query
CACHE
Replication
CACHE
LOGGING
DB replication
GATEWAY
Microservice1
Microservice2
Microservice3
R
R
W/R
R
R
replica DB
Replication
LEADER DB
Replication
replica DB