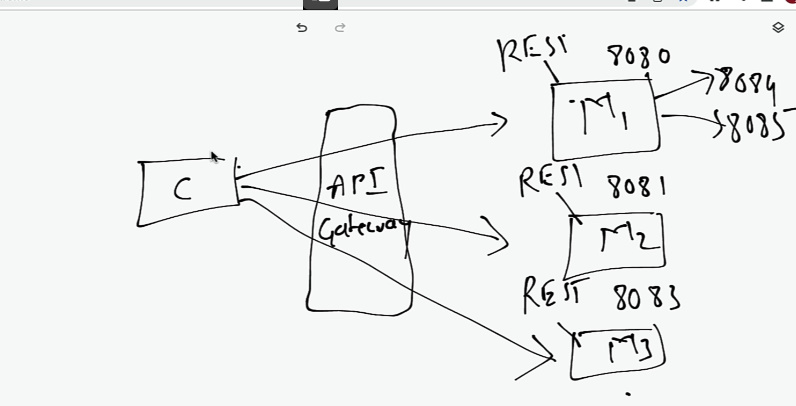


Chiar daca Service Registry permite la microservices sa comunice intre ele fara a fi necesar sa hard codam host si port, mai ramane problema cum client sa comunince cu REST API oferite de microservices

* Putem avea mii de REST API si client inca trebuie sa le memoreze host si port
* **API Gateway** sta intre Microservicii si Client



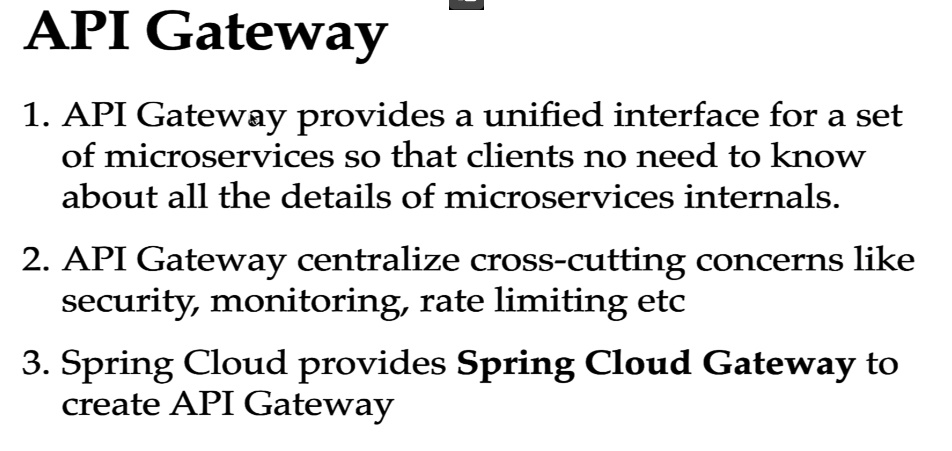
El il va ajuta pe client prin faptul ca il va scuti din a memora host si port la fiecare Service. El se va asigura sa obtina host si port necesar pentru client.

* Deci API Gateway are 3 roluri:

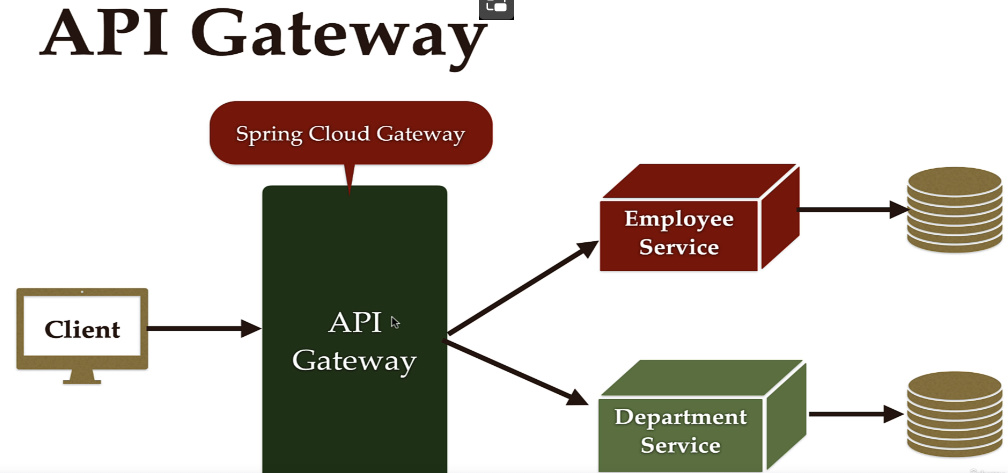
- Route request

- Load Balacing, poate face si asta

- Security – in loc sa adugam security la fiecare microservice, adaugam doar la API Gateway. Nu va trebuie sa se autentifice in fiecare, ci doar in gatway



**Implementare**



1. **Cream un microservice(project separat) ce va fi API GateWay**

ne trebuie Spring GateWay ca dependency si Eureka Client, caci acest GateWat tot va fi client la eureka server, dar si actuators ar fi bine de pus

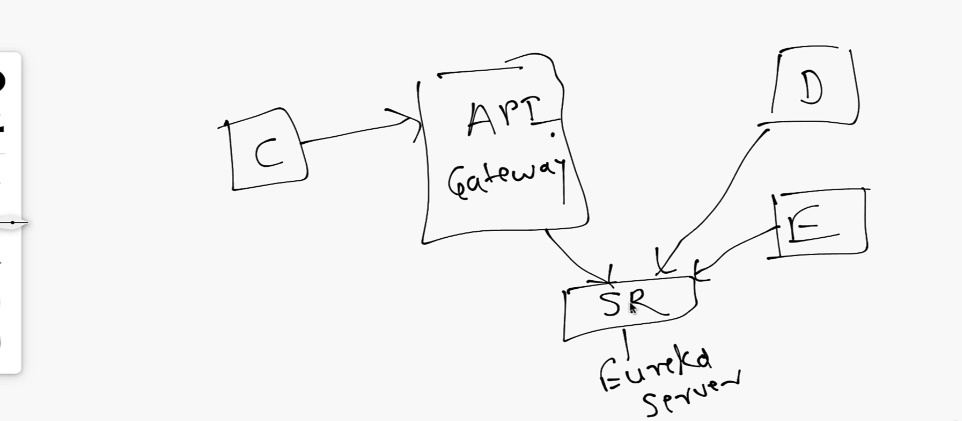
1. **Inregistram API Gateway ca Eureka Client in Eureka Server**

server.port=9191  
  
spring.application.name=API-GATEWAY  
  
eureka.client.service-url.defaultZone = http://localhost:8761/eureka/

1. **Configuram API Gateway Routes**

**Route** – preluarea requestului de la client si trimiterea lui la microservice necesar, fara a fi necesar ca client sa introduce host:port

* Pentru a descoperi host si port la microservices, gatway va folosi eureka server, adica Service Registry, caci doar gateway e eureka client si poate comunica cu service registry, adica eureka server



* Putem configra routes in 2 moduri:

- prin properties(mai des folosit)

- prin code

spring.cloud.gateway.routes[0].id=EMPLOYEE-SERVICE  
spring.cloud.gateway.routes[0].uri=lb://EMPLOYEE-SERVICE  
spring.cloud.gateway.routes[0].predicates[0] = Path=/api/employees

**spring.cloud.gateway.routes[0].id** – adauga numele la microservice asa cum e el inregistsrat si in Eureka server

**spring.cloud.gateway.routes[0].uri** – aici punem url pentru load balancer, si il punem sub forma lb://NUME-SERVICE

**spring.cloud.gateway.routes[0].predicates[0]** – punem aici care e link din rest controller de la microservice inregistsrat, adica EMPLOYEE-SERVICE, care vrem sa il putem folosi direct din gateway

* Deci, desi gayeway are port 9191, si EMPLOYEE-SERVICE 8081, nu va trebui sa folosim localhost:8081/api/employees, ci direct folosim port si host la gateway microservice, adica localhost:9091/api/employees, si el se va uita in properties la predicades la ce microservice corespunde acest path, apoi va folosi eureka server pentru a prelua ip la microservice si port si va returna rezultatul

spring.cloud.gateway.routes[0].id=EMPLOYEE-SERVICE  
spring.cloud.gateway.routes[0].uri=lb://EMPLOYEE-SERVICE  
spring.cloud.gateway.routes[0].predicates[0] = Path=/api/employees/\*\*  
  
spring.cloud.gateway.routes[1].id = DEPARTMENT-SERVICE  
spring.cloud.gateway.routes[1].uri = lb://DEPARTMENT-SERVICE  
spring.cloud.gateway.routes[1].predicates[0] = /api/departments/\*\*

/\*\* inseamna ca dupa acel path mai putem avea orice inca

**Create automatically routes**

Nu e prea comod sa tot scriem de mana id, uri si predicates la microservices. O putem insa automatiza.

spring.cloud.gateway.discovery.locator.enabled=true  
spring.cloud.gateway.discovery.locator.lower-case-service-id=true  
logging.level.org.springframework.cloud.gateway.handler.RoutePredicateHandlerMapping = DEBUG

* spring.cloud.gateway.discovery.locator.enabled – activeaza discovery, adica ca gateway sa scaneze pe baza la eureka server microserviciile
* spring.cloud.gateway.discovery.locator.lower-case-service-id – va transforma numele la microservicii in lower case, ca sa fie toate identice la sigur
* Logarea doar arata logguri pentru routes mapping facute

Acum atentie! In URL la gateway mereu vom folosi si numele la service!

[~~localhost:9191/api/employees/1~~](http://localhost:9191/api/employees/1)

[localhost:9191/employee-service/api/employees/1](http://localhost:9191/employee-service/api/employees/1)

si vedem ca employee-service e in lower case!