CS590

Software Architecture

Auxiliary Data Input Service

Supervision by

Professor Siamak Tavakoli

Maharishi International University

Team

Md Rashedul Bari
Md Rana Hossain
Somal Chakraborty
Md Shah Jalal Mazumder
Amit Yadav
Yasmin Beyene Belay

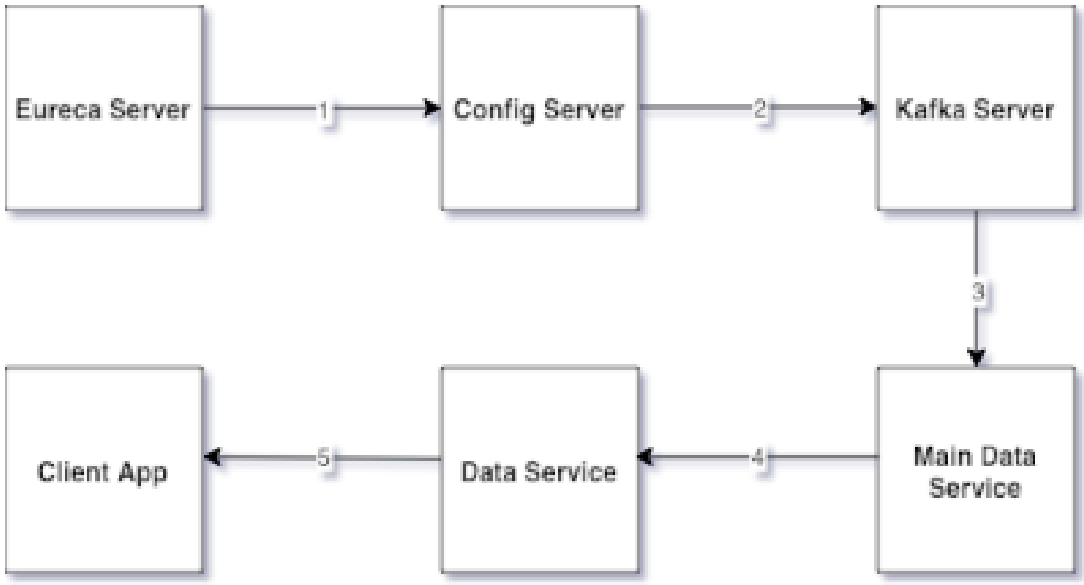


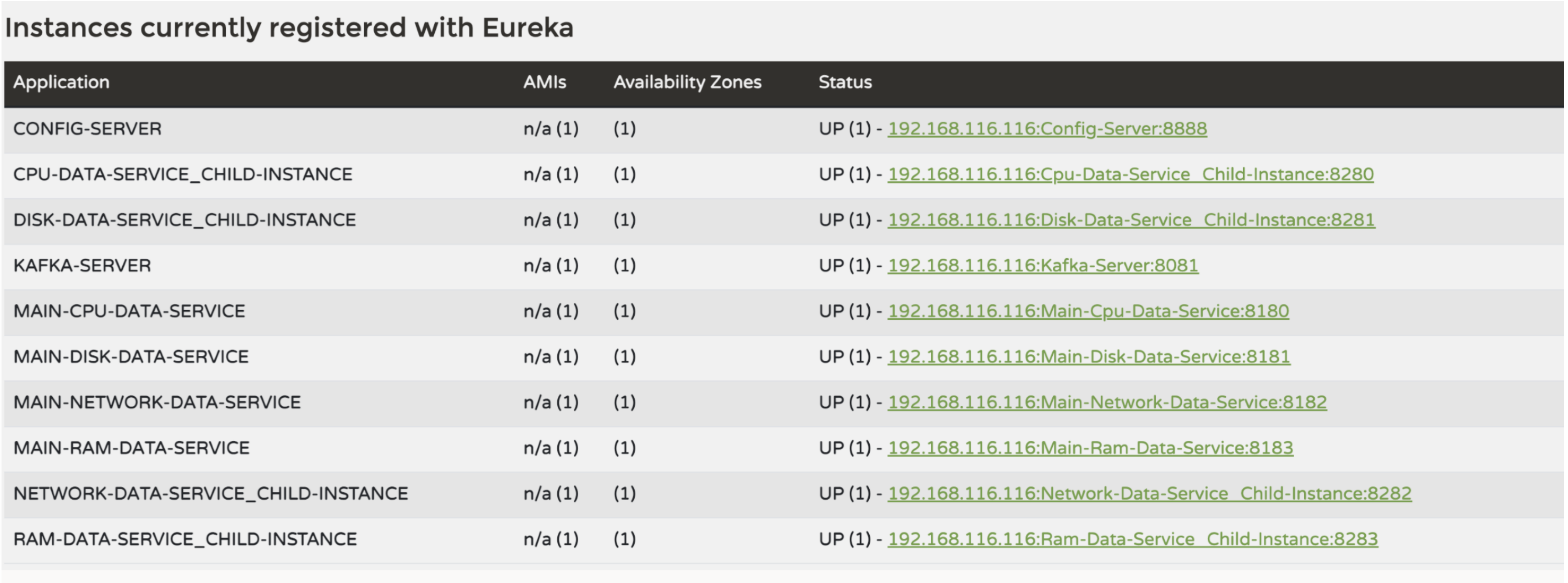
Diagram: Services to run sequentially

Service Sequence

Prerequisites

Netdata Service
Zookeeper Service
Kafka Service

Eureka Service Registry



Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
CONFIG-SERVER	n/a (1)	(1)	UP (1) - 192.168.116.116:Config-Server:8888
CPU-DATA-SERVICE_CHILD-INSTANCE	n/a (1)	(1)	UP (1) - 192.168.116.116:Cpu-Data-Service_Child-Instance:8280
DISK-DATA-SERVICE_CHILD-INSTANCE	n/a (1)	(1)	UP (1) - 192.168.116.116:Disk-Data-Service_Child-Instance:8281
KAFKA-SERVER	n/a (1)	(1)	UP (1) - 192.168.116.116:Kafka-Server:8081
MAIN-CPU-DATA-SERVICE	n/a (1)	(1)	UP (1) - 192.168.116.116:Main-Cpu-Data-Service:8180
MAIN-DISK-DATA-SERVICE	n/a (1)	(1)	UP (1) - 192.168.116.116:Main-Disk-Data-Service:8181
MAIN-NETWORK-DATA-SERVICE	n/a (1)	(1)	UP (1) - 192.168.116.116:Main-Network-Data-Service:8182
MAIN-RAM-DATA-SERVICE	n/a (1)	(1)	UP (1) - 192.168.116.116:Main-Ram-Data-Service:8183
NETWORK-DATA-SERVICE_CHILD-INSTANCE	n/a (1)	(1)	UP (1) - 192.168.116.116:Network-Data-Service_Child-Instance:8282
RAM-DATA-SERVICE_CHILD-INSTANCE	n/a (1)	(1)	UP (1) - 192.168.116.116:Ram-Data-Service_Child-Instance:8283

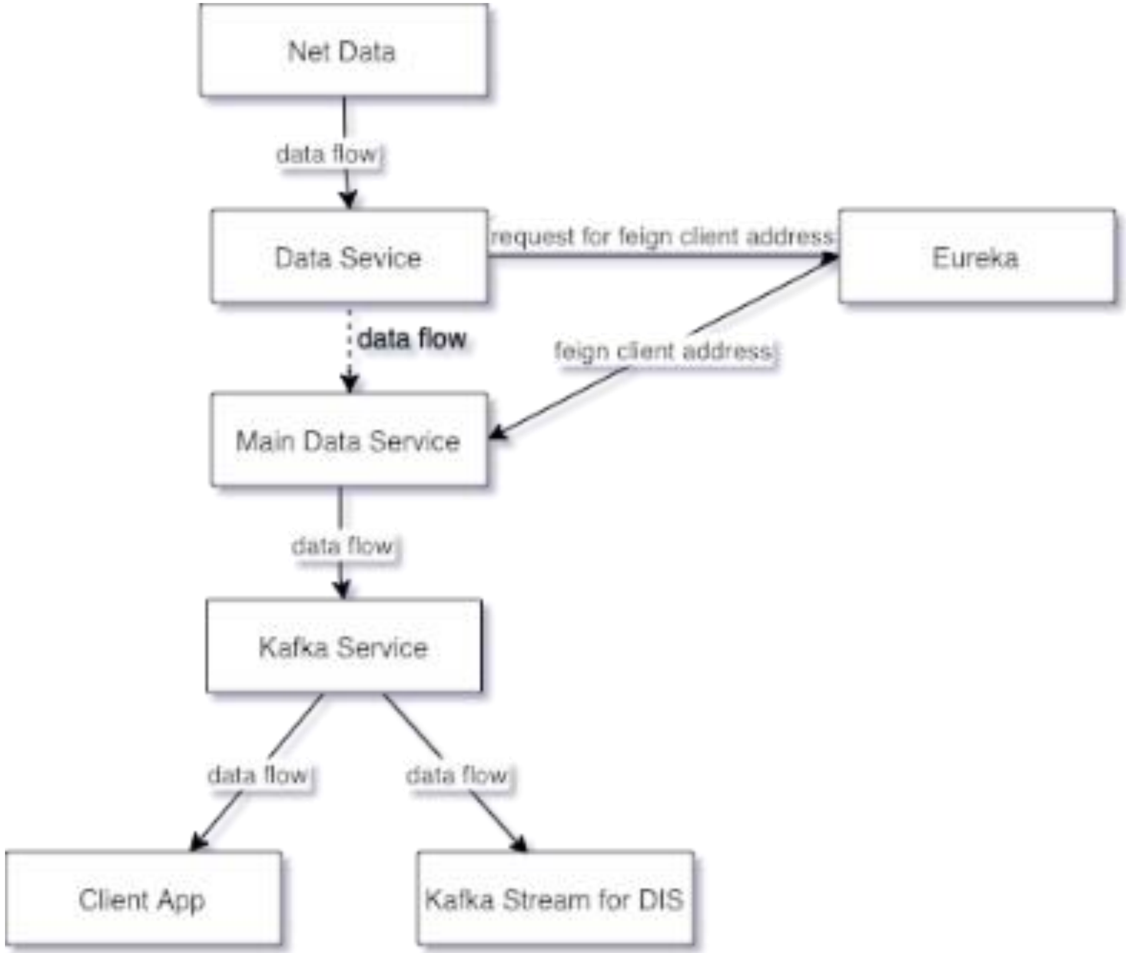
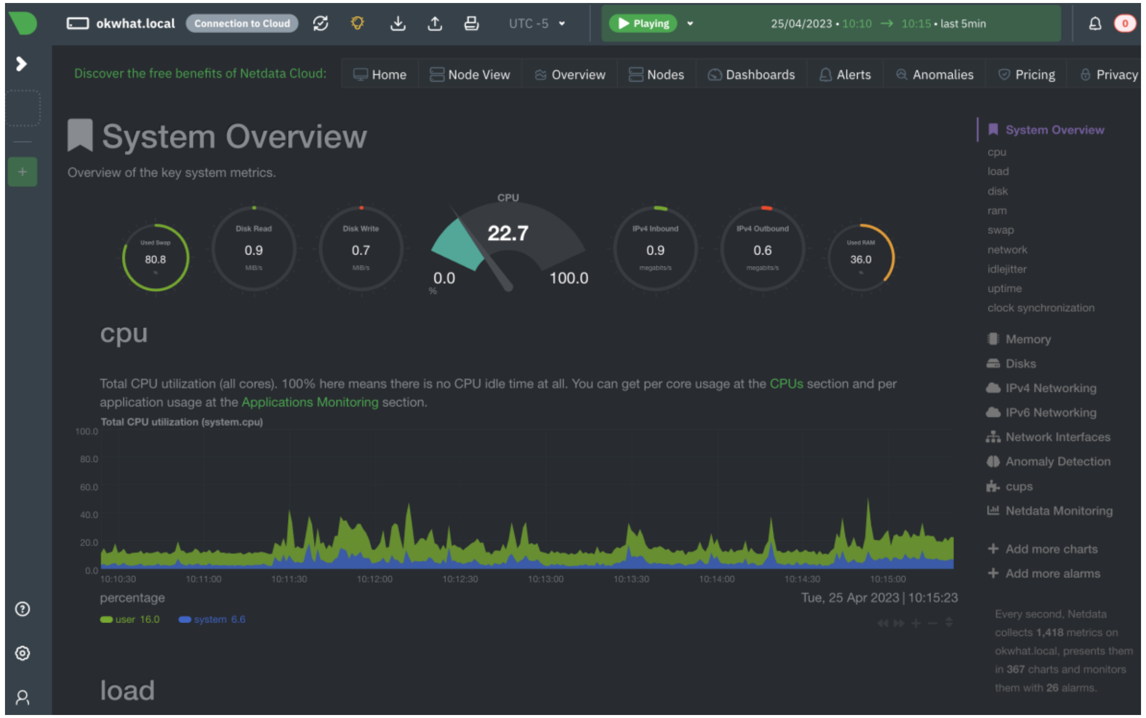


Diagram: Data Flow Chart

Data Flow Diagram

Retrieving And Sending Machine Metrics To Kafka Stream

Netdata



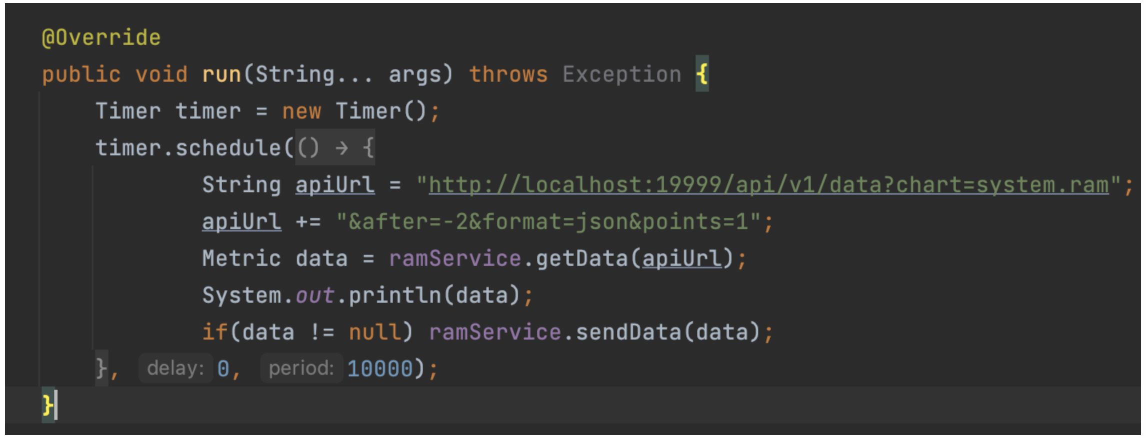
Total CPU utilization (all cores). 100% here means there is no CPU idle time at all. You can get per core usage at the CPUs section and per application usage at the Applications Monitoring section.

Every second, Netdata collects 1,418 metrics on okwhat.local, presents them in 367 charts and monitors them with 26 alarms.

We are using Netdata which is a lightweight, real-time monitoring tool which is collecting system metrics, including RAM, CPU, disk usage and network data.

Url: http://localhost:19999

MetricService Fetching Data

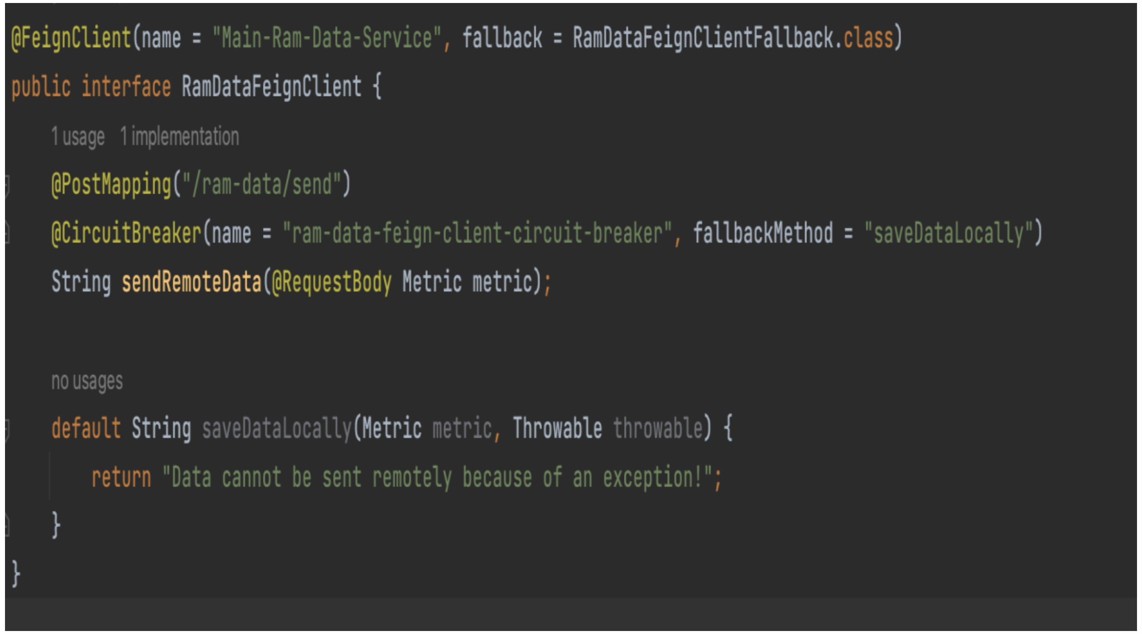


```
@Override
public void run(String... args) throws Exception {
    Timer timer = new Timer();
    timer.schedule(() → {
            String apiUrl = "http://localhost:19999/api/v1/data?chart=system.ram";
            apiUrl += "&after=-2&format=json&points=1";
            Metric data = ramService.getData(apiUrl);
            System.out.println(data);
            if(data != null) ramService.sendData(data);
    }, delay: 0, period: 10000);
}
```

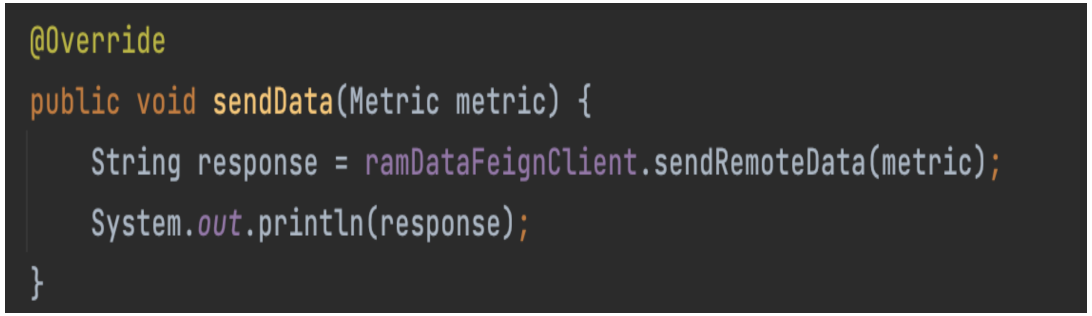
MetricService e.g. RamService fetches data from netdata and proceeds to send it

Url:
http://localhost:19999/api/v1/data?chart=system.ram&after=-2&format=json&points=1

MetricService Sending Data To MainMetricsService



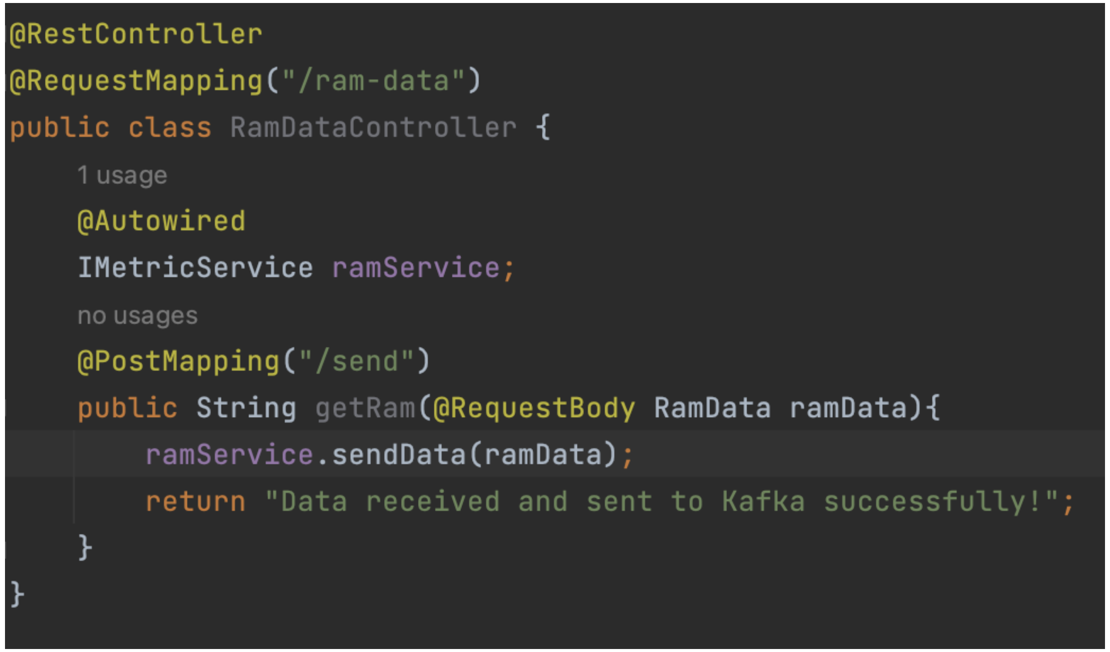
```
@FeignClient(name = "Main-Ram-Data-Service", fallback = RamDataFeignClientFallback.class)
public interface RamDataFeignClient {
    1 usage   1 implementation
    @PostMapping("/ram-data/send")
    @CircuitBreaker(name = "ram-data-feign-client-circuit-breaker", fallbackMethod = "saveDataLocally")
    String sendRemoteData(@RequestBody Metric metric);

    no usages
    default String saveDataLocally(Metric metric, Throwable throwable) {
        return "Data cannot be sent remotely because of an exception!";
    }
}
```



```
@Override
public void sendData(Metric metric) {
    String response = ramDataFeignClient.sendRemoteData(metric);
    System.out.println(response);
}
```

MainMetricsService e.g. MainRamService is registered on eureka and is a feign client for RamService.

MainRamService has a POST api that receives the metrics from RamService

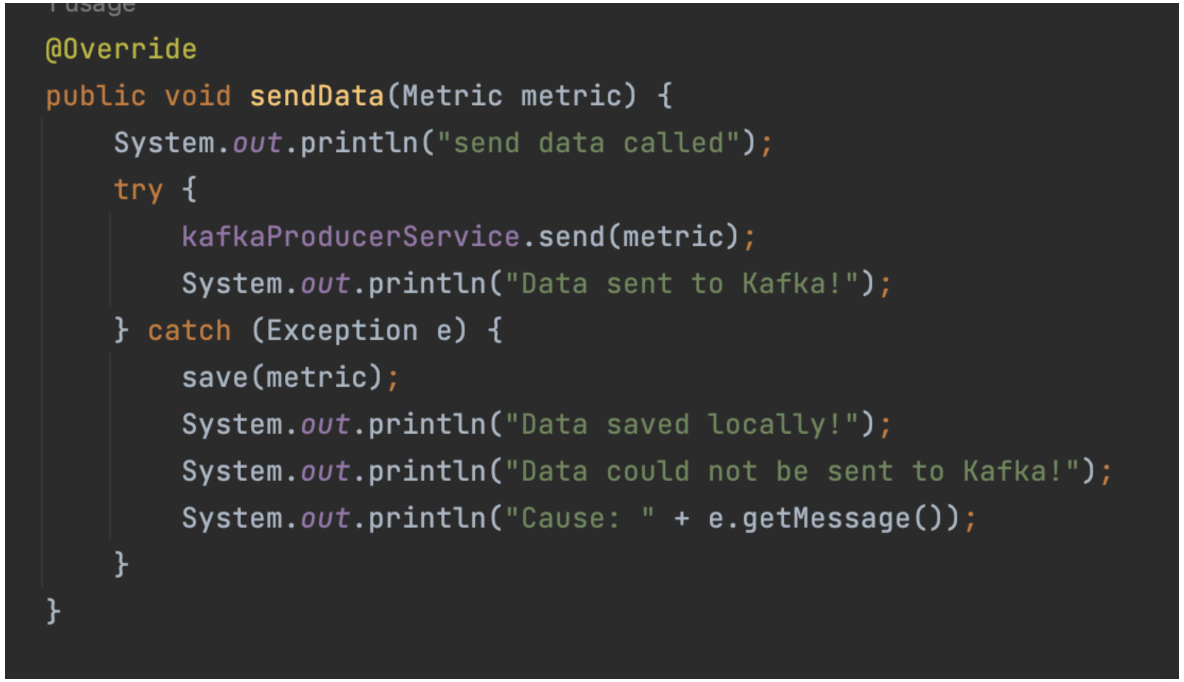


```
@RestController
@RequestMapping("/ram-data")
public class RamDataController {
    1 usage
    @Autowired
    IMetricService ramService;
    no usages
    @PostMapping("/send")
    public String getRam(@RequestBody RamData ramData){
        ramService.sendData(ramData);
        return "Data received and sent to Kafka successfully!";
    }
}
```

MainMetricsService Receives Data From MetricService

MainMetricsService receives the metrics on the exposed API **/ram-data/send** via Rest Call

MainMetricsService Sends Data To Kafka Stream

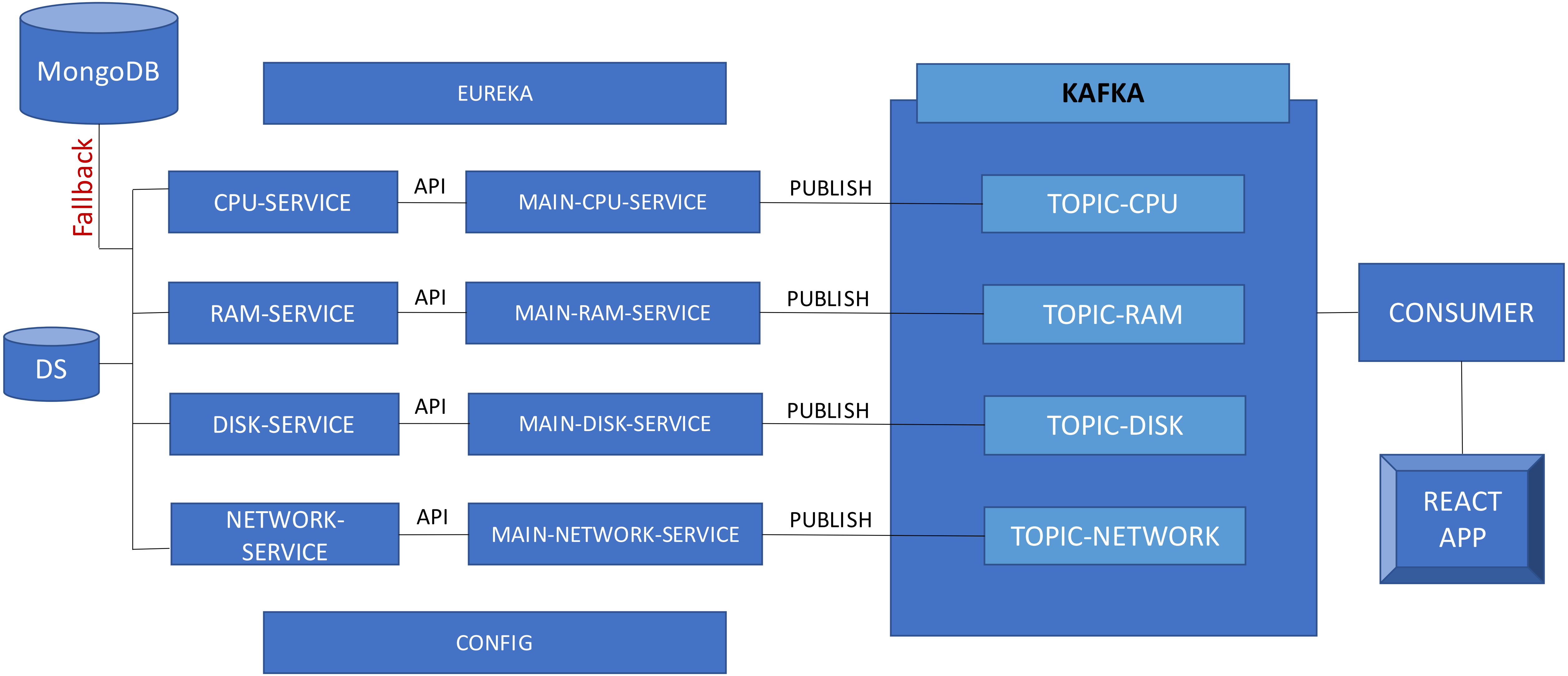


```
1 usage
@Override
public void sendData(Metric metric) {
    System.out.println("send data called");
    try {
        kafkaProducerService.send(metric);
        System.out.println("Data sent to Kafka!");
    } catch (Exception e) {
        save(metric);
        System.out.println("Data saved locally!");
        System.out.println("Data could not be sent to Kafka!");
        System.out.println("Cause: " + e.getMessage());
    }
}
```

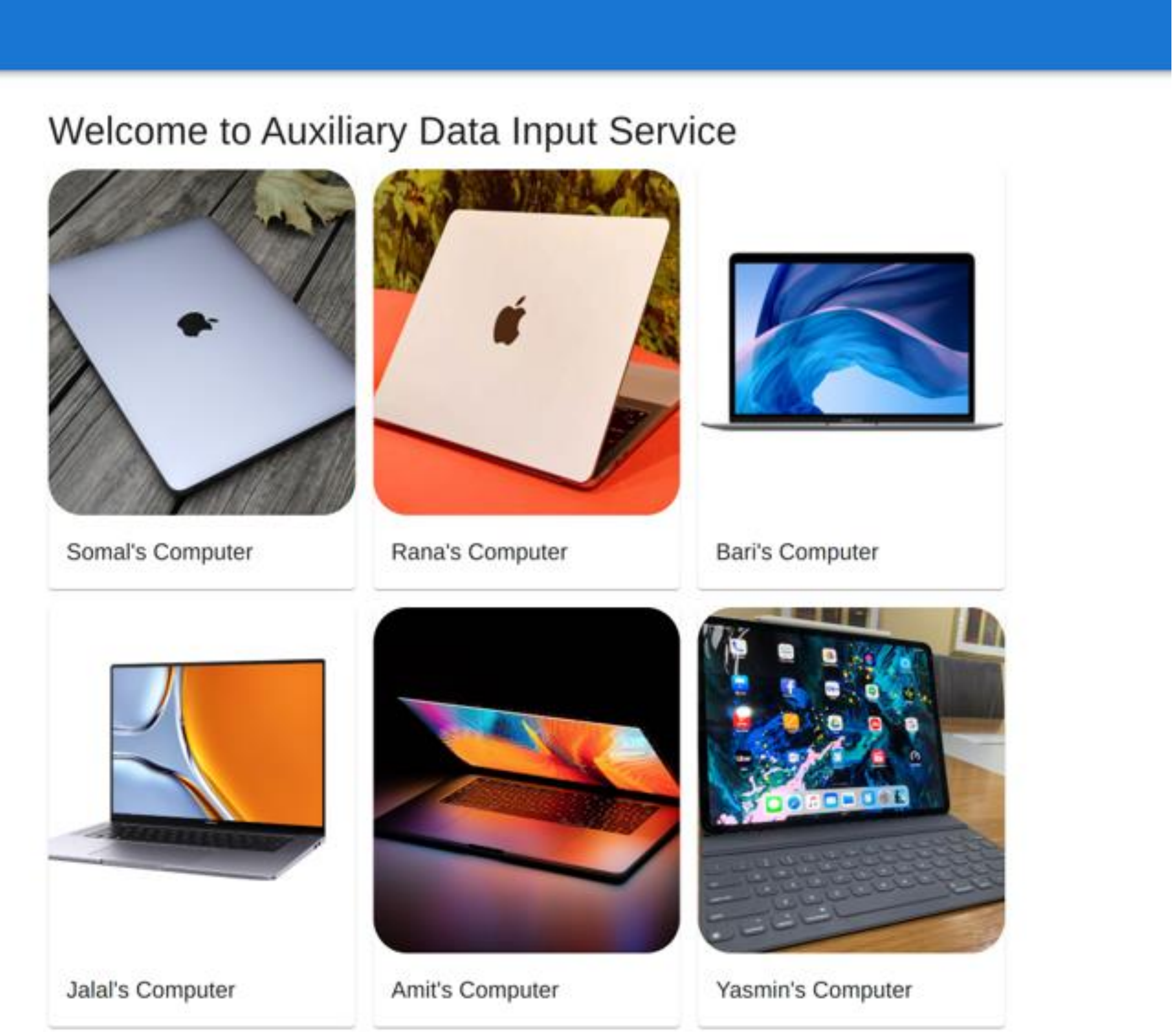
MainMetricsService upon receiving the metrics sends it to the kafka stream on the relevant topic (Topic_Ram-1 here)

Kafka Consumer To Integrate All Metrics

AUXILIARY DATA INPUT SERVICES



ADIS Client Application

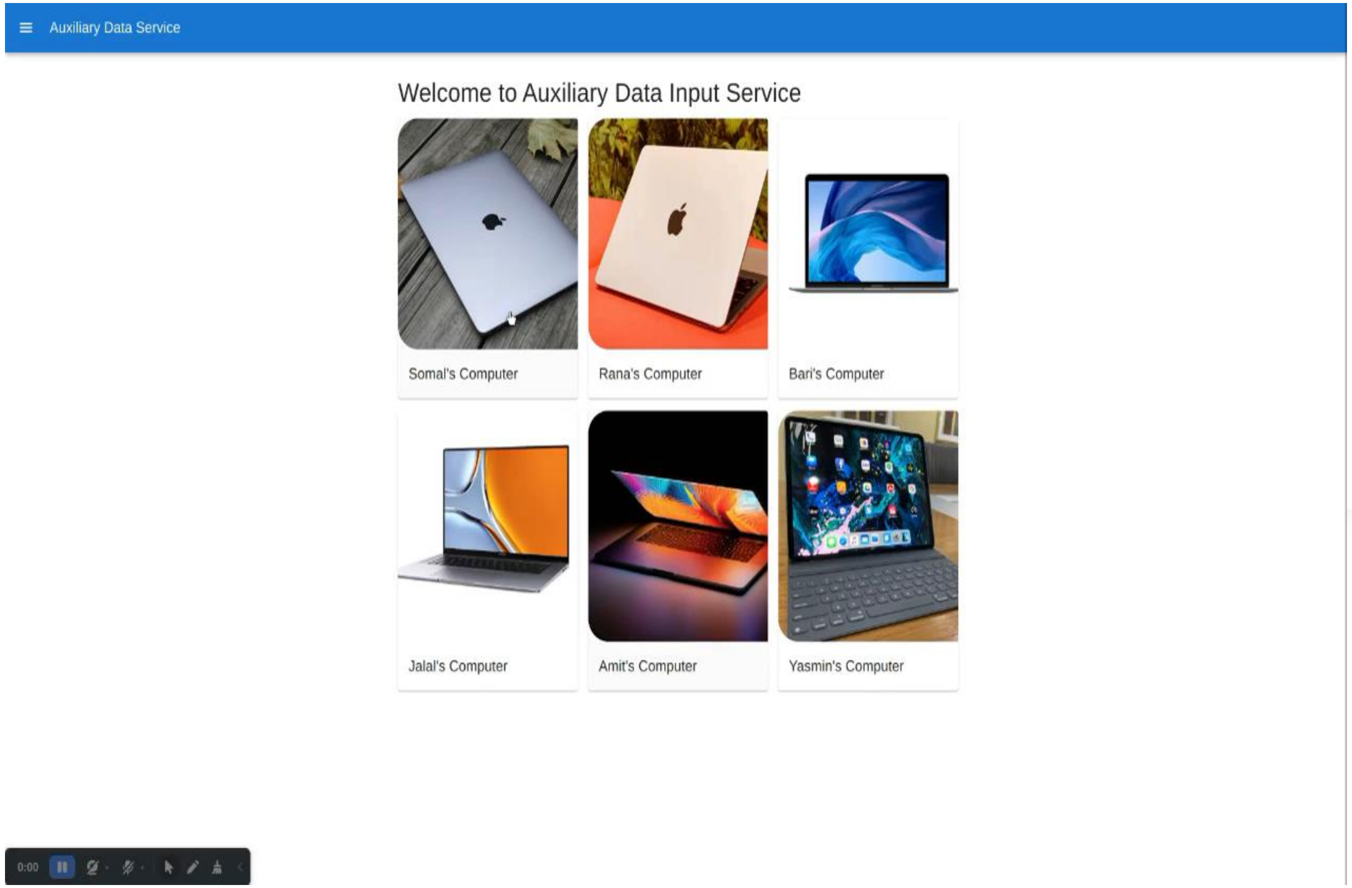


Welcome to Auxiliary Data Input Service

Real time data visualization

Tools: React

Protocol: RestAPI

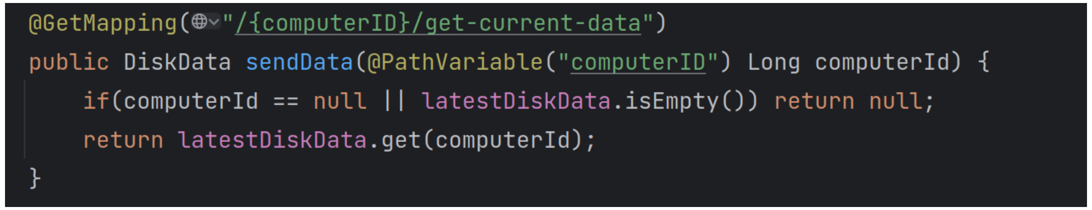


Welcome to Auxiliary Data Input Service

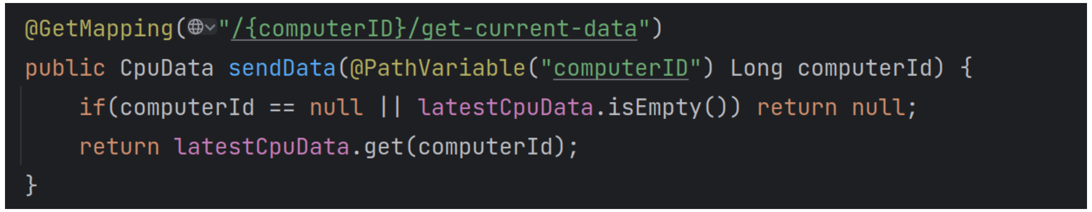
Demo of Client Application

Shows different computer:

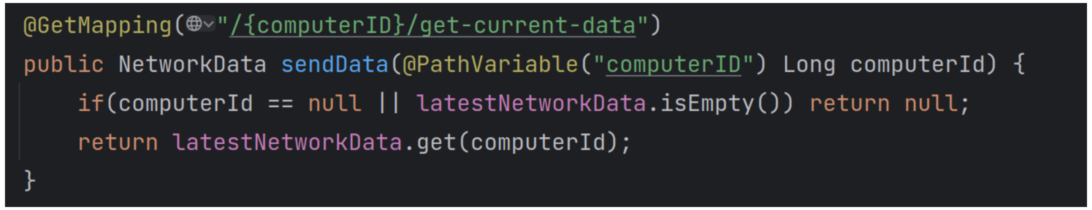
1. Ram Data
2. CPU Data
3. Disk Data
4. Network Data



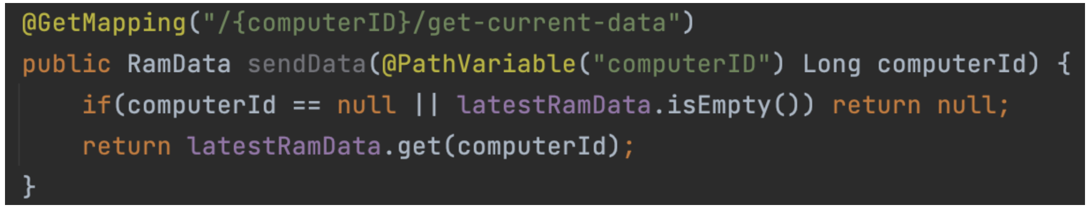
```
@GetMapping("/{computerID}/get-current-data")
public DiskData sendData(@PathVariable("computerID") Long computerId) {
    if(computerId == null || latestDiskData.isEmpty()) return null;
    return latestDiskData.get(computerId);
}
```



```
@GetMapping("/{computerID}/get-current-data")
public CpuData sendData(@PathVariable("computerID") Long computerId) {
    if(computerId == null || latestCpuData.isEmpty()) return null;
    return latestCpuData.get(computerId);
}
```



```
@GetMapping("/{computerID}/get-current-data")
public NetworkData sendData(@PathVariable("computerID") Long computerId) {
    if(computerId == null || latestNetworkData.isEmpty()) return null;
    return latestNetworkData.get(computerId);
}
```



```
@GetMapping("/{computerID}/get-current-data")
public RamData sendData(@PathVariable("computerID") Long computerId) {
    if(computerId == null || latestRamData.isEmpty()) return null;
    return latestRamData.get(computerId);
}
```

ADIS RestAPI for Client

4 API to get bellow data:

- Ram
- CPU
- Disk
- Network

Improvement of the legacy code

Thank you

Any Question?