



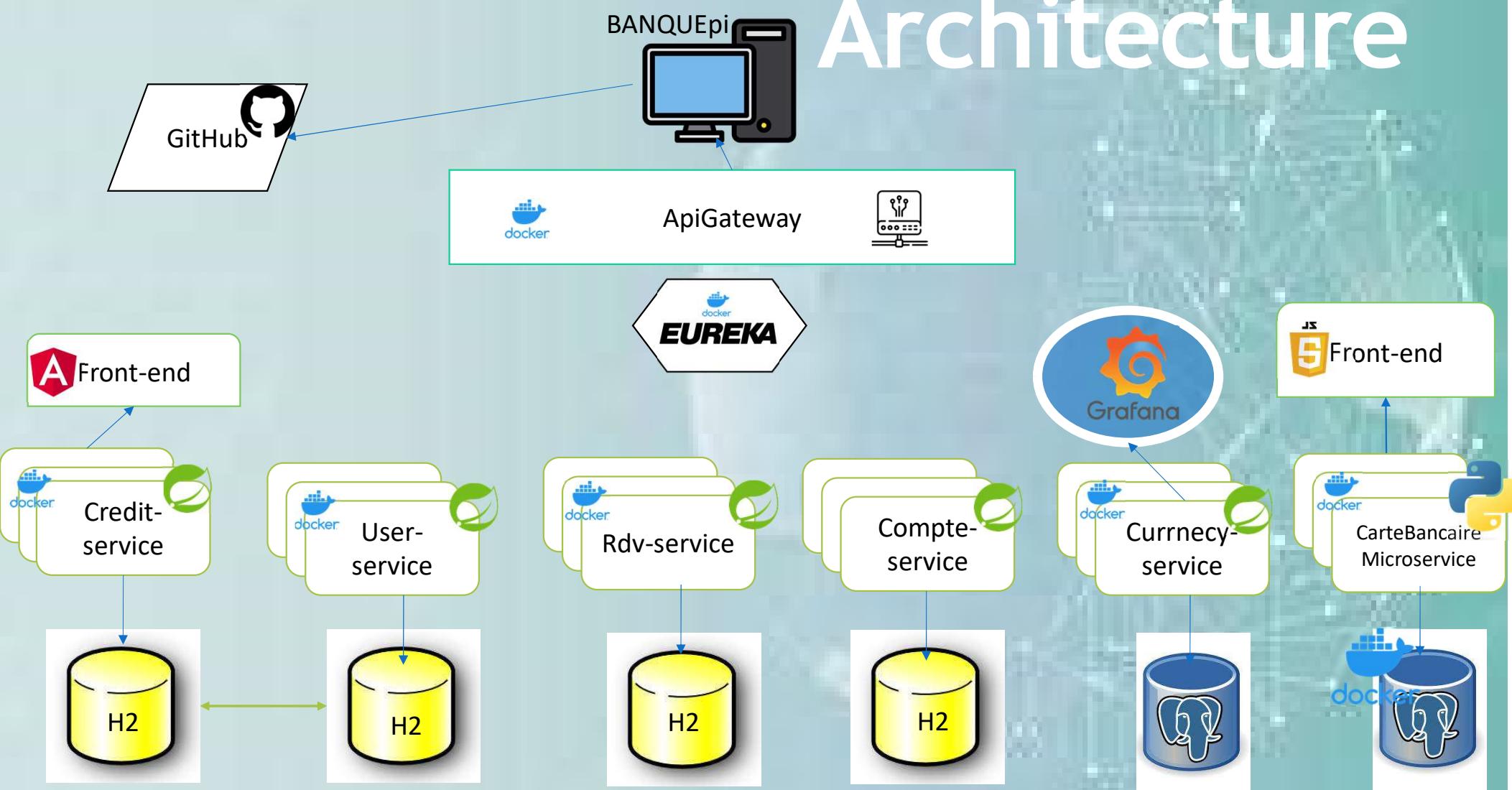
DEFINITION



Microservice

Nous avons développé les microservices suivants : « User », « Crédit », « Compte », « Carte », « Curancy » et « RDV ». Ces microservices correspondent à différentes fonctionnalités de notre système, permettant la gestion des utilisateurs, des crédits, des comptes, des cartes bancaires, de la conversion de devises et des rendez-vous. Chacun de ces microservices joue un rôle spécifique dans l'architecture globale de votre application, contribuant ainsi à la mise en place d'un système complet et fonctionnel.

Architecture





Back-end

<http://localhost:8761/eureka>

EUREKA

Nous avons intégré le backend de tous nos microservices en utilisant Eureka.
Eureka est utilisé comme service de découverte et d'enregistrement des instances de nos microservices, permettant ainsi une communication efficace entre eux

The screenshot shows the Eureka System Status interface at localhost:8761/eureka. The top banner features the word "EUREKA" in large, bold, black letters against a blue background with a network graph pattern.

System Status

Environment	test	Current time	2023-06-24T02:38:34 +0000
Data center	default	Uptime	00:36
		Lease expiration enabled	true
		Renews threshold	10
		Renews (last min)	16

DS Replicas

localhost			
Instances currently registered with Eureka			
Application	AMIs	Availability Zones	Status
COMPTE-SERVICE	n/a (1)	(1)	UP (1) - 36d3c47fc5f1:compte-service:8081
CREDIT-SERVICE	n/a (1)	(1)	UP (1) - TNLT1412.vermeg.com:credit-service:8085
GATEWAY-SERVICE	n/a (1)	(1)	UP (1) - 044d5e3635bb:gateway-service:8762
RENDEZVOUS-SERVICE	n/a (1)	(1)	UP (1) - TNLT1412.vermeg.com:rendezVous-service:8083
USER-SERVICE	n/a (1)	(1)	UP (1) - Zed4a0a12152:user-service:8082

[http://localhost:8762/](http://localhost:8762)

Gateway



Tous nos microservices sont opérationnels et disponibles via le gateway. Le gateway sert de point d'entrée unique pour l'accès à nos microservices, offrant une interface centralisée pour les requêtes entrantes et sortantes. Cela permet de simplifier la gestion des requêtes et d'améliorer l'efficacité de l'architecture globale.

Project Explorer

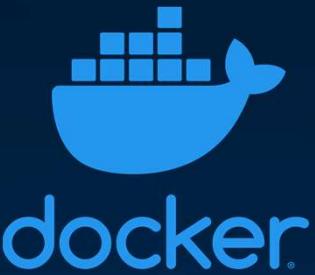
- > credit-Service [boot] [devtools] [finalProject aya2]
- > discovery-service [boot] [finalProject aya2]
- > gateway-service [boot] [devtools] [finalProject aya2]
- > src/main/java
 - > application.yml
- > JRE System Library [JavaSE-11]
- > Maven Dependencies
- > target/generated-sources/annotations
- > src
 - > target
 - > dockerfile
 - > mvnw
 - > mvnw.cmd
 - > pom.xml
- > gestionCompte [boot] [finalProject aya2]
- > rendezvouss-service [boot] [devtools] [finalProject aya2]
- > user-service [boot] [devtools] [finalProject aya2]

```
application.yml
```

```
server:
  port: 8762
spring:
  application:
    name: gateway-service
  main:
    web-application-type: reactive
  cloud:
    gateway:
      globalcors:
        cors-configurations:
          '/**':
            allowed-origins: "*"
            allowed-methods: "*"
            allowed-headers: "*"
      routes:
        - id: credit
          uri: lb://credit-service
          predicates:
            - Path=/api/simulator/**
        - id: user
          uri: lb://user-service
          predicates:
```

localhost:8762/api/users/

```
[{"id":1,"organizationId":1,"siteId":1,"username":"sayed","password":"$2a$12$V2IuTBSRrf7aRW6rQd9Ng06.Y8gMohamed","email":"sayed@test.com","phone":"010123456789","position":"Manager","roles":[{"id":1,"name":"Manager"}],"createdBy":1,"modifiedBy":1,"deleted":false,"lastModified":1625000000000,"version":1}, {"id":2,"organizationId":1,"siteId":2,"username":"ahmed","password":"$2a$12$FU.Wpyz4sReCNUG06ht.IufVkJ2Mahmoud","email":"ahmed@teat.com","phone":"01097654321","position":"Developer","roles":[{"id":1,"name":"Developer"}],"createdBy":1,"modifiedBy":1,"deleted":false,"lastModified":1625000000000,"version":1}, {"id":3,"organizationId":2,"siteId":3,"username":"mohammad","password":"$2a$12$fICaEBqntTuxoQb6nm8kmmOSHjAli","email":"mohammad@test.com","phone":"010678954321","position":"Analyst","roles":[{"id":1,"name":"Analyst"}],"createdBy":1,"modifiedBy":1,"deleted":false,"lastModified":1625000000000,"version":1}, {"id":4,"organizationId":3,"siteId":4,"username":"hamada","password":"$2a$12$lHkg5NDV6qJ9HKCWSVxgV.gnRQgIbrahim","email":"hamada@test.com","phone":"010531247896","position":"Developer","roles":[{"id":1,"name":"Developer"}],"createdBy":1,"modifiedBy":1,"deleted":false,"lastModified":1625000000000,"version":1}, {"id":5,"organizationId":2,"siteId":3,"username":"tamer","password":"$2a$12$nX328Dw5DBr9/6WFIBP9neGRZqjHassen","email":"tamer@test.com","phone":"010975874321","position":"Analyst","roles":[{"id":1,"name":"Analyst"}],"createdBy":1,"modifiedBy":1,"deleted":false,"lastModified":1625000000000,"version":1}, {"id":6,"organizationId":1,"siteId":5,"username":"admin","password":"$2a$12$0UG2jkLjP1QD76DxGzPot.060000Mohamed","email":"sayed@test.com","phone":"010123456789","position":"Manager","roles":[{"id":1,"name":"Manager"}]}]
```



Docker est une plateforme open-source qui permet de créer et de déployer des applications dans des conteneurs légers et isolés. Cela facilite la gestion des applications, la portabilité et l'isolation de leurs dépendances.

```
ancaireMs.py index.html dockerfile

# Use the official Python image as the base image
FROM python:3.8-slim-buster

# Set the working directory in the container
WORKDIR /app

# Copy the requirements.txt file to the container
COPY requirements.txt .

# Install the Python dependencies
RUN pip install --no-cache-dir -r requirements.txt

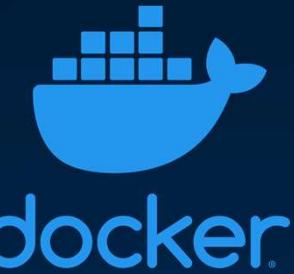
# Copy the application code to the container
COPY . .

# Expose the port on which the application will run
EXPOSE 5000

# Run the application
CMD ["python", "CarteBancaireMs.py"]
```

```
version: "3.0"

services:
  eurekaserver:
    build: discovery-service
    container_name: eurekaserver
    image: eurekaserver
    networks:
      - eureka-server
    depends_on:
      - fluent-bit
    ports:
      - "8761:8761"
    logging:
      driver: fluentd
      options:
        fluentd-address: localhost:24225
        tag: eurekaserver
    microservice-gestion-compte:
      build: gestionCompte
      container_name: gestionCompte
      # network_mode: host
      image: msgestioncompte
      ports:
        - "8081:8081"
      networks:
        - eureka-server
      environment:
        - eureka.client.serviceUrl.defaultZone=http://eurekaserver:8761/eureka
      depends_on:
        - eurekaserver
        - fluent-bit
      logging:
        driver: fluentd
        options:
          fluentd-address: localhost:24225
          tag: gestionCompte
    microservice-gestion-user:
      build: user-service
      container_name: user-service
      # network_mode: host
      image: msgestionusers
      ports:
        - "8082:8082"
      networks:
        - eureka-server
      environment:
        - eureka.client.serviceUrl.defaultZone=http://eurekaserver:8761/eureka
      depends_on:
        - eurekaserver
        - fluent-bit
      logging:
        driver: fluentd
        options:
          fluentd-address: localhost:24225
          tag: user-service
```



Nous avons dokarisé la base de données, Eureka, le gateway et les microservices.

```
version: '3'  
services:  
  loki:  
    image: grafana/loki:latest  
    container_name: loki  
    expose:  
      - "3100"  
    networks:  
      - loki  
    volumes:  
      - loki-volume:/loki  
  
  grafana:  
    image: grafana/grafana:latest  
    container_name: grafana  
    ports:  
      - "3000:3000"  
    environment:  
      GF_RENDERING_SERVER_URL: http://renderer:8081/render  
      GF_RENDERING_CALLBACK_URL: http://grafana:3000/  
      GF_LOG_FILTERS: rendering:debug  
    networks:  
      - loki  
    volumes:  
      - grafana-volume:/var/lib/grafana  
  
  renderer:  
    image: grafana/grafana-image-renderer:latest  
    container_name: grafana-image-renderer  
    expose:  
      - "8081"  
    environment:  
      ENABLE_METRICS: "true"  
    networks:  
      - loki  
    volumes:  
      - renderer-volume:/usr/share/grafana-image-renderer  
  
networks:  
  loki:  
    external: true  
  
volumes:  
  grafana-volume:  
  loki-volume:  
  renderer-volume:
```

```
version: '3.0'  
services:  
  postgres-14.3:  
    image: "postgres:14.3-alpine"  
    container_name: "postgres14ms"  
    environment:  
      LANG: C.UTF-8  
      LC_ALL: C.UTF-8  
      POSTGRES_USER: postgres  
      POSTGRES_PASSWORD: Postgres+123  
      POSTGRES_DB: cartecredit  
    ports:  
      - "5432:5432"  
    volumes:  
      - "./scripts:/docker-entrypoint-initdb.d"  
    command: ["--max_prepared_transactions=100", "--max_connections=500"]  
    healthcheck:  
      test: ["CMD-SHELL", "pg_isready -U postgres"]  
      interval: 10s  
      retries: 10  
    networks:  
      - eureka-server  
  
  carte-bancaire-ms:  
    build:  
      context: .  
      dockerfile: Dockerfile  
    container_name: carte-bancaire-ms  
    image: carte-bancaire-ms  
    ports:  
      - "5000:5000"  
    depends_on:  
      - postgres-14.3  
    networks:  
      - eureka-server  
  
networks:  
  eureka-server:  
    driver: bridge
```



```
cmd C:\Windows\System32\cmd.exe - docker-compose up

eurekaserver | 2023-06-24 02:28:21.131 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
eurekaserver | 2023-06-24 02:29:21.131 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
eurekaserver | 2023-06-24 02:30:21.131 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
eurekaserver | 2023-06-24 02:31:21.131 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
eurekaserver | 2023-06-24 02:32:17.983 INFO 1 --- [trap-executor-0] c.n.d.s.r.aws.ConfigClusterResolver : Resolving eureka endpoints via configuration
eurekaserver | 2023-06-24 02:32:20.511 INFO 1 --- [thresholdUpdater] c.n.e.r.PeerAwareInstanceRegistryImpl : Current renewal threshold is : 6
eurekaserver | 2023-06-24 02:32:21.132 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
eurekaserver | 2023-06-24 02:32:32.940 INFO 1 --- [trap-executor-0] c.n.d.s.r.aws.ConfigClusterResolver : Resolving eureka endpoints via configuration
eurekaserver | 2023-06-24 02:33:21.132 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
eurekaserver | 2023-06-24 02:34:21.132 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
eurekaserver | 2023-06-24 02:35:21.132 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
eurekaserver | 2023-06-24 02:36:21.133 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
eurekaserver | 2023-06-24 02:37:17.644 INFO 1 --- [nio-8761-exec-7] c.n.e.registry.AbstractInstanceRegistry : Registered instance CREDIT-SERVICE/TNLT1412.vermeg.com:credit-service:8085 with s
tatus UP (replication=false)
gateway-service | 2023-06-24 02:37:17.984 INFO 1 --- [trap-executor-0] c.n.d.s.r.aws.ConfigClusterResolver : Resolving eureka endpoints via configuration
eurekaserver | 2023-06-24 02:37:18.155 INFO 1 --- [nio-8761-exec-8] c.n.e.registry.AbstractInstanceRegistry : Registered instance CREDIT-SERVICE/TNLT1412.vermeg.com:credit-service:8085 with s
tatus UP (replication=true)
eurekaserver | 2023-06-24 02:37:21.132 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
gestioncompte | 2023-06-24 02:37:24.371 INFO 1 --- [trap-executor-0] c.n.d.s.r.aws.ConfigClusterResolver : Resolving eureka endpoints via configuration
user-service | 2023-06-24 02:37:32.941 INFO 1 --- [trap-executor-0] c.n.d.s.r.aws.ConfigClusterResolver : Resolving eureka endpoints via configuration
eurekaserver | 2023-06-24 02:38:21.133 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
eurekaserver | 2023-06-24 02:38:26.574 INFO 1 --- [nio-8761-exec-2] c.n.e.registry.AbstractInstanceRegistry : Registered instance RENDEZVOUS-SERVICE/TNLT1412.vermeg.com:rendezVous-service:808
3 with status UP (replication=false)
eurekaserver | 2023-06-24 02:38:27.088 INFO 1 --- [nio-8761-exec-3] c.n.e.registry.AbstractInstanceRegistry : Registered instance RENDEZVOUS-SERVICE/TNLT1412.vermeg.com:rendezVous-service:808
3 with status UP (replication=true)
eurekaserver | 2023-06-24 02:39:21.133 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
eurekaserver | 2023-06-24 02:40:21.133 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
eurekaserver | 2023-06-24 02:41:21.134 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
eurekaserver | 2023-06-24 02:42:17.985 INFO 1 --- [trap-executor-0] c.n.d.s.r.aws.ConfigClusterResolver : Resolving eureka endpoints via configuration
eurekaserver | 2023-06-24 02:42:21.133 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
gestioncompte | 2023-06-24 02:42:24.373 INFO 1 --- [trap-executor-0] c.n.d.s.r.aws.ConfigClusterResolver : Resolving eureka endpoints via configuration
user-service | 2023-06-24 02:42:32.942 INFO 1 --- [trap-executor-0] c.n.d.s.r.aws.ConfigClusterResolver : Resolving eureka endpoints via configuration
user-service | 2023-06-24 02:43:04.798 INFO 1 --- [nio-8082-exec-2] o.a.c.c.[Tomcat].[localhost].[]/ : Initializing Spring DispatcherServlet 'dispatcherServlet'
user-service | 2023-06-24 02:43:04.799 INFO 1 --- [nio-8082-exec-2] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
user-service | 2023-06-24 02:43:04.818 INFO 1 --- [nio-8082-exec-2] o.s.web.servlet.DispatcherServlet : Completed initialization in 19 ms
user-service | 2023-06-24 02:43:04.993 INFO 1 --- [nio-8082-exec-2] o.h.h.i.QueryTranslatorFactoryInitiator : HHH000397: Using ASTQueryTranslatorFactory
user-service | User.server.port: 8082
eurekaserver | 2023-06-24 02:43:21.134 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
eurekaserver | 2023-06-24 02:44:21.134 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
eurekaserver | 2023-06-24 02:45:21.134 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
eurekaserver | 2023-06-24 02:46:21.135 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
eurekaserver | 2023-06-24 02:47:17.986 INFO 1 --- [trap-executor-0] c.n.d.s.r.aws.ConfigClusterResolver : Resolving eureka endpoints via configuration
eurekaserver | 2023-06-24 02:47:20.511 INFO 1 --- [thresholdUpdater] c.n.e.r.PeerAwareInstanceRegistryImpl : Current renewal threshold is : 10
eurekaserver | 2023-06-24 02:47:21.136 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
gestioncompte | 2023-06-24 02:47:32.943 INFO 1 --- [trap-executor-0] c.n.d.s.r.aws.ConfigClusterResolver : Resolving eureka endpoints via configuration
user-service | 2023-06-24 02:48:21.135 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
eurekaserver | 2023-06-24 02:49:21.135 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
```



On a utilisé Grafana pour visualiser et analyser les logs de notre système. Grafana est une plateforme de surveillance et de visualisation qui vous permet de collecter et de présenter des données en temps réel sous forme de tableaux de bord interactifs



Communication Microservice

Nous avons connecté les deux microservices "Utilisateur" et "Crédit". Lorsque le service de demande de crédit reçoit une demande, il vérifie d'abord si l'utilisateur existe en appelant le microservice "Utilisateur" via le service d'ajout d'utilisateurs. Si l'utilisateur n'existe pas, le service d'ajout d'utilisateurs créera un nouvel utilisateur. Dans le cas où l'utilisateur existe déjà, le service de demande de crédit procède directement à la demande de crédit.

The image shows two Postman requests side-by-side. The left request is a GET to `http://localhost:8054/api/users`, which returns a JSON response with two user objects. The right request is a POST to `http://localhost:8054/api/credit-requests/1`, which also returns a JSON response with a credit request object.

Left Request (GET to /users):

```
1 {
2   "id": 1,
3   "organizationId": null,
4   "siteId": null,
5   "username": null,
6   "password": "00485",
7   "name": "FATTI",
8   "email": "ayaF@gmail.com",
9   "phone": null,
10  "position": null,
11  "roles": []
12 },
13 {
14   "id": 2,
15   "organizationId": null,
16   "siteId": null,
17   "username": null,
```

Right Request (POST to /credit-requests/1):

```
1 {
2   "id": 6,
3   "cin": "9522510",
4   "job": "inge",
5   "civilState": "celeb",
6   "creationDate": "2022-07-13T13:47:20.079+00:00",
7   "address": "tunis",
8   "creditRequestStatus": "CREATED",
9   "requesterName": "ayaf"
```



Front-end

Gestion carte bancaire

Nous avons créé une application CRUD (Create, Read, Update, Delete) pour la gestion des cartes bancaires en utilisant Python et JavaScript. Dokarisé la base de données en PostgreSQL, et l'application

Jupyter CarteBancaireMS Dernière Sauvegarde : il y a 11 heures (auto-sauvegardé)

File Edit View Insert Cell Kernel Widgets Help

Entrée []:

```
from flask import Flask, jsonify, request, render_template
from flask_cors import CORS
import psycopg2

app = Flask(__name__)
CORS(app)

# Database connection parameters
db_params = {
    'host': '172.27.0.2',
    'port': 5432,
    'database': 'cartecredit',
    'user': 'postgres',
    'password': 'Postgres+123'
}

# Connect to the PostgreSQL database
conn = psycopg2.connect(**db_params)
cursor = conn.cursor()

# Create the cards table if it doesn't exist
create_table_query = """
CREATE TABLE IF NOT EXISTS cards (
    id SERIAL PRIMARY KEY,
    card_number VARCHAR(16),
    card_holder VARCHAR(50),
    expiry_date DATE
);
```

Card List

Card Number	Card Holder	Expiry Date
1235	maissa ben romdhan	14/15
124	145	14/09
147852369	aya fathallah	14/09
147852333	amel fathallah	14/12

Add Card

Card Number:

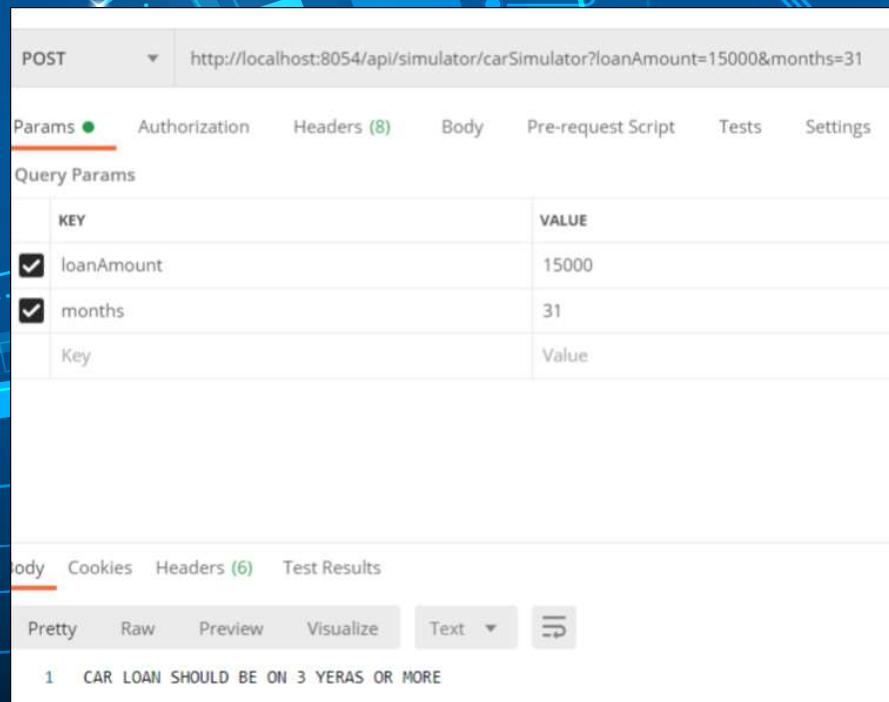
Card Holder:

Expiry Date:

Add Card

Simulation credit

Nous avons développé une interface front-end en utilisant Angular pour les services de simulation. L'appel vers le microservice se fait via le gateway qui est déjà opérationnel.



POST <http://localhost:8054/api/simulator/carSimulator?loanAmount=15000&months=31>

Params Authorization Headers (8) Body Pre-request Script Tests Settings

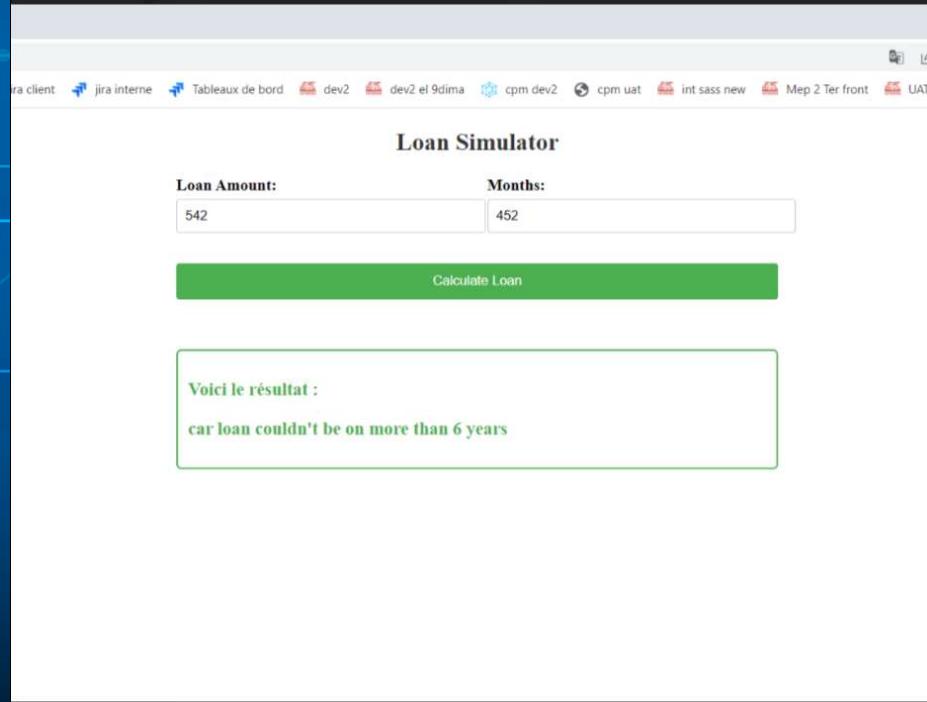
Query Params

KEY	VALUE
<input checked="" type="checkbox"/> loanAmount	15000
<input checked="" type="checkbox"/> months	31

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize Text

1 CAR LOAN SHOULD BE ON 3 YERAS OR MORE



Loan Simulator

Loan Amount: 542 Months: 452

Calculate Loan

Voici le résultat :
car loan couldn't be on more than 6 years



Conclusion



Conclusion

En conclusion, nous avons développé les Microservices "user", "Crédit", "Compte", "Carte", "Curancy" et "RDV" pour gérer différentes fonctionnalités de votre système. Cette approche basée sur les microservices vous permet d'organiser votre application de manière modulaire, favorisant la scalabilité et la maintenance indépendante de chaque composant.