



الجمهورية العربية السورية

جامعة دمشق

كلية الهندسة المعلوماتية

قسم هندسة البرمجيات ونظم المعلومات

الوظيفة الثانية

LMS with Microservices

براء محمد الفتال

عباده احمد المقداد

مالك مأمون القداح

محمد حسين ذياب

رؤى عصام البصيري

باشراف

م. أمير أبو الشعر

م. آية معطي

- قمنا بتطوير نظام تعليم الكتروني يهدف الى تسهيل عملية التعلم حيث لدينا ثلاث أنواع من المستخدمين (مسؤول مدرب متعلم)
- يقدم النظام العديد من الخدمات المقسمة على الشكل
 1. قسم للمستخدمين وادارتهم
 2. قسم للدورات التعليمية
 3. قسم الامتحانات
 4. قسم الدفع

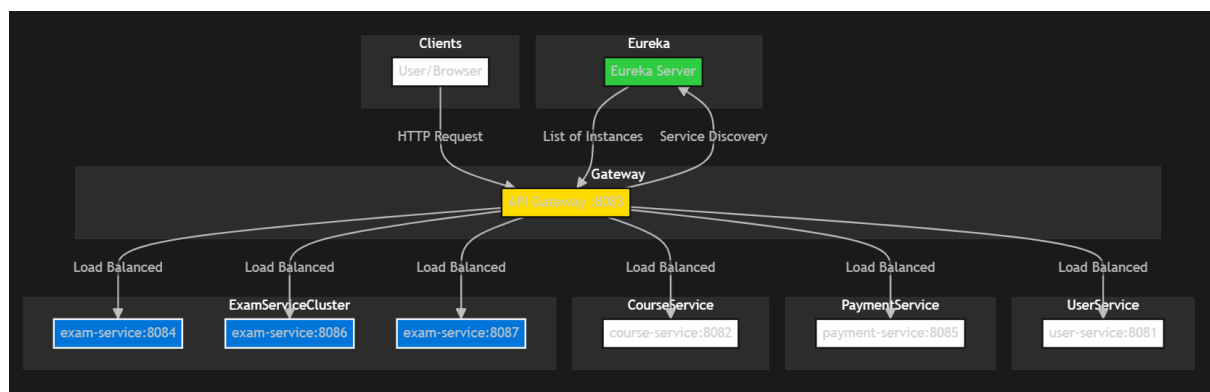
قمنا بإضافة API Gateway لضمان توجيه جميع طلبات المستخدمين عبر بوابة موحدة و service discovery لعنونة الخدمات بشكل ديناميكي

فكانت هيكلية النظام على الشكل

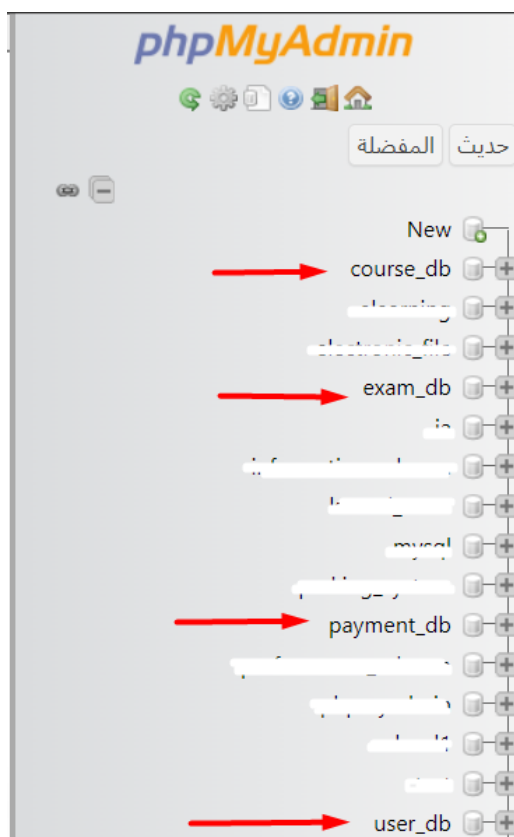
DS Replicas

Instances currently registered with Eureka

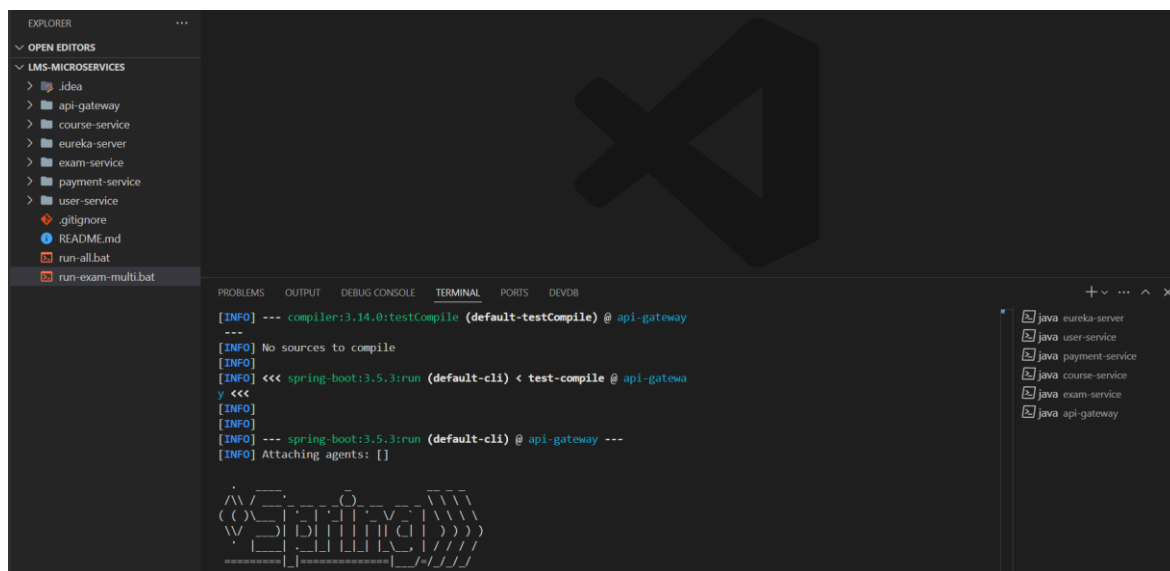
Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (١)	(١)	UP (١) - api-gateway:8083
COURSE-SERVICE	n/a (١)	(١)	UP (١) - course-service:8082
EXAM-SERVICE	n/a (٣)	(٣)	UP (٣) - localhost:exam-service:8086 , localhost:exam-service:8087 , localhost:exam-service:8084
PAYMENT-SERVICE	n/a (١)	(١)	UP (١) - localhost:payment-service:8085
USER-SERVICE	n/a (١)	(١)	UP (١) - user-service:8081



حيث كل خدمة متصلة بقاعدة بيانات منفصلة خاصة بها



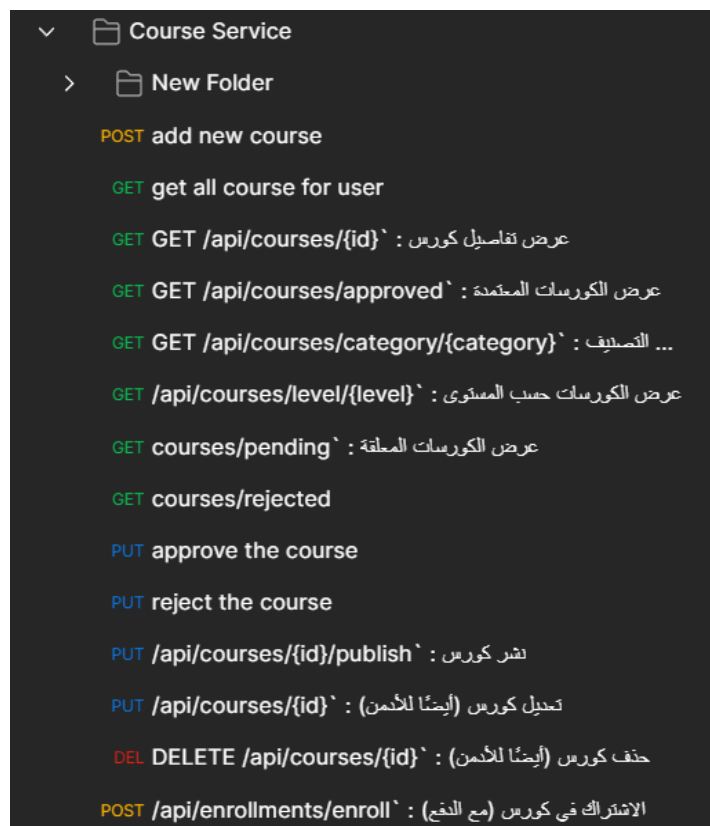
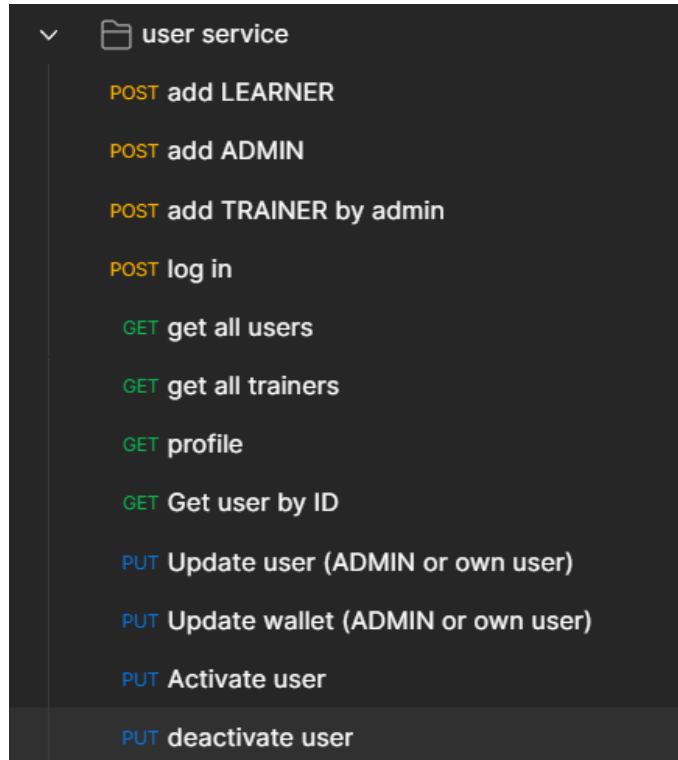
وهذه بنية المشروع مع طريقة التشغيل






رابط المشروع على git hub

<https://github.com/ObadaAlmogdad/microservices>

نعرض الان بعض الصور من postman لعرض بعض api المتاحة في كل service



- ▼  pymant service
 - POST pay
 - POST refund
 - POST charge
 - GET get all payments
- ▼  exam service
 - ▼  resalt
 - POST add results
 - PUT update results
 - DEL delete results
 - GET results
 - GET results by id
 - GET results for user
 - GET results for exam
 - POST add exam
 - GET get exam for course
 - POST add questions
 - GET get questions
 - POST add answers for LEARNER
 - PUT evaluate answers

قمنا بإضافة API Gateway لضمان توجيه جميع طلبات المستخدمين عبر بوابة موحدة

```
application.yml X
api-gateway > src > main > resources > application.yml
1  spring:
2    application:
3      name: api-gateway
4    cloud:
5      gateway:
6        discovery:
7          locator:
8            enabled: true
9            lower-case-service-id: true
10     routes:
11 >       - id: user-service ...
17 >       - id: course-service ...
23 >       - id: eureka-server ...
29 >       - id: payment-service ...
35 >       - id: exam-service ...
41
42   server:
43     port: 8083
44
45 > eureka: ...
54 |
55 > globalcors: ...
```

service discovery لعنونة الخدمات بشكل ديناميكي

```
eureka-server > src > main > resources > application.yml
1  spring:
2    application:
3      name: eureka-server
4
5  server:
6    port: 8761
7
8  eureka:
9    instance:
10     hostname: localhost
11    client:
12     register-with-eureka: false
13     fetch-registry: false
14     service-url:
15       defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/
16   server:
17     wait-time-in-ms-when-sync-empty: 0
```

بالنسبة لمعالجة حالات فشل الاتصال وتأخر الاستجابة بين الخدمات

تم استخدام المكتبة

```
# Resilience4j Configuration
resilience4j:
  circuitbreaker:
    instances:
      enrollmentClient:
        sliding-window-size: 10
        minimum-number-of-calls: 5
        failure-rate-threshold: 50
        wait-duration-in-open-state: 5000
        permitted-number-of-calls-in-half-open-state: 3
  retry:
    instances:
      enrollmentClient:
        max-attempts: 3
        wait-duration: 1000
  timelimiter:
    instances:
      enrollmentClient:
        timeout-duration: 3000
```

حيث

sliding-window-size: 10 حجم نافذة المراقبة (عدد الطلبات الأخيرة التي سيتم مراقبتها لتحديد نسبة الفشل)

minimum-number-of-calls: 5 أقل عدد من الطلبات يجب أن يتم قبل أن يبدأ Circuit Breaker في تقييم نسبة الفشل.

failure-rate-threshold: 50 النسبة المئوية للفشل هنا (50%)، إذا تجاوزت هذه النسبة في النافذة، يتم فتح الدائرة (Circuit Breaker) يوقف الطلبات مؤقتًا

wait-duration-in-open-state: 5000 المدة بالمللي ثانية التي يبقى فيها Circuit Breaker في حالة "مفتوحة" (لا يسمح بالطلبات) قبل أن يحاول إعادة الطلبات

permitted-number-of-calls-in-half-open-state: 3 عدد الطلبات المسموح بها أثناء حالة نصف مفتوحة - لاختبار إذا كانت الخدمة عادت للعمل هنا 3 طلبات

retry إعدادات إعادة المحاولة (Retry)، أي كم مرة يعيد النظام محاولة الطلب عند الفشل.

instances نفس فكرة الـ circuitbreaker، يمكن تخصيص الإعدادات لكل عميل.

max-attempts: 3 أقصى عدد محاولات للطلب (المحاولة الأولى + محاولتين إضافيتين = 3 محاولات)

wait-duration: 1000 المدة بالمللي ثانية (بين كل محاولة)

timeout-duration: 3000 أقصى مدة (بالمللي ثانية) يُسمح بها للطلب (هنا 3 ثواني)، إذا تجاوزها الطلب يعتبر فاشلاً.

بالنسبة ل Load Balancing يتم على الشكل التالي

1. تسجيل الخدمات في Eureka

- كل خدمة (user-service, payment-service, course-service, exam-service, ...) تسجل نفسها في Eureka Server باسم منطقي (مثلاً: exam-service).
- يمكن تشغيل أكثر من نسخة (instance) من نفس الخدمة على منافذ مختلفة (مثلاً: exam-service : 8084, 8086, 8087).

2. اكتشاف الخدمات (Service Discovery)

- أي خدمة أو Gateway تريد التواصل مع خدمة أخرى، لا تحتاج معرفة عنوانها أو منفذها.
- فقط تستخدم الاسم المنطقي مثلاً lb://exam-service أو: @FeignClient(name = "exam-service").
- Eureka يعطي قائمة بكل الـ instances النشطة لهذه الخدمة.

3. توزيع الحمل تلقائيًا (Load Balancing)

- عند كل طلب جديد يتم اختيار instance واحدة من القائمة Round Robin.
- Spring Cloud LoadBalancer هو المسؤول عن توزيع الحمل.
- كل طلب قد يذهب إلى instance مختلفة، مما يوزع الضغط ويزيد من الاعتمادية.

تم الاختبار على exam service

```
1 @echo off
2
3 REM على المنفذ 8084 instance exam-service شغل
4 start cmd /k "cd exam-service && mvn spring-boot:run -Dspring-boot.run.arguments=--server.port=8084"
5 timeout /t 5
6
7 REM على المنفذ 8086 instance exam-service شغل
8 start cmd /k "cd exam-service && mvn spring-boot:run -Dspring-boot.run.arguments=--server.port=8086"
9 timeout /t 5
10
11 REM على المنفذ 8087 instance exam-service شغل
12 start cmd /k "cd exam-service && mvn spring-boot:run -Dspring-boot.run.arguments=--server.port=8087"
```

عند الطلبات المتكررة على الخدمة يتم تنفيذها على instance مختلفة مما يوزع الضغط ويزيد من الاعتمادية