



Système Universitaire **SOA**

Architecture Orientée
Services

Presenter par :
Hamzaoui Moetez
Saidane Med Lamine



Sommaire

01

CONTEXTE ET
OBJECTIFS

03

SERVICES REST

05

PÉRIMÈTRE
FONCTIONNEL

07

SÉCURITÉ - JWT

07

AVANTAGES DE
L'ARCHITECTURE
SOA

02

ARCHITECTURE
GLOBALE

04

SERVICES SOAP

06

STACK
TECHNIQUE

08

DÉPLOIEMENT
DOCKER

08

CONCLUSION



Contexte et objectifs du projet



Contexte :

L'université souhaite moderniser son système d'information en adoptant une architecture orientée services (SOA) pour séparer les fonctionnalités en services indépendants.



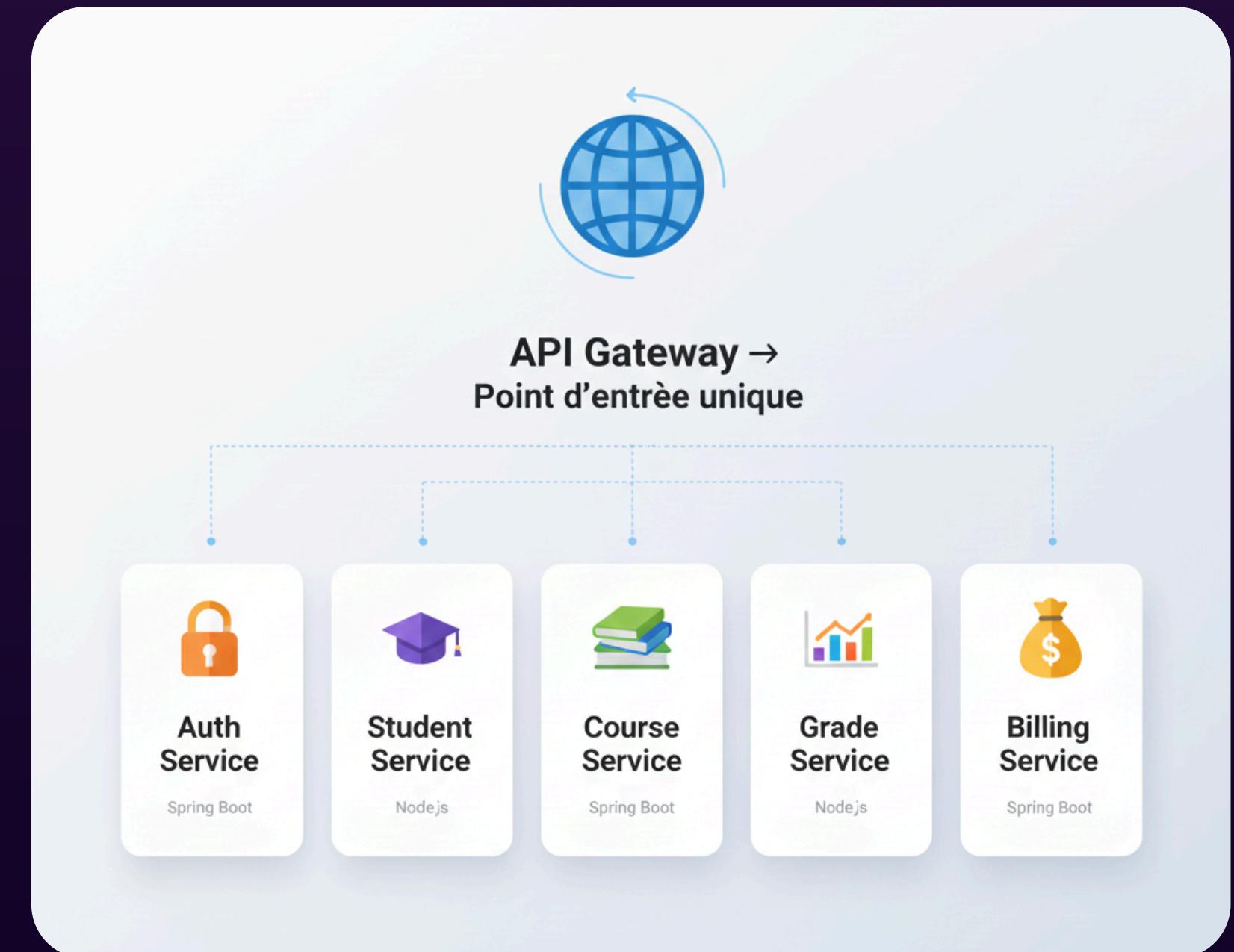
Solution proposée :

- ✓ Mettre en place une architecture SOA complète
- ✓ Développer des services web hétérogènes (REST + SOAP)
- ✓ Assurer la sécurité via JWT
- ✓ Garantir l'interopérabilité entre technologies
- ✓ Préparer le déploiement conteneurisé (Docker)

Architecture globale du système

Architecture globale du système :

- ✓ Architecture SOA basée sur 6 services
- ✓ API Gateway comme point d'entrée unique
- ✓ Communication via REST & SOAP



Services REST

Auth Service



- Tech:
 - Spring Boot + JWT
- Endpoints:
 - POST /auth/register
 - POST /auth/login
 - GET /auth/me

Student Service



- Tech:
 - Node.js / Express
- Endpoints:
 - GET /students
 - POST /students
 - PUT /students/:id
 - DELETE /students/:id

Grade Service



- Tech:
 - Node.js / Express
- Endpoints:
 - POST /grades
 - GET /grades/student/:id
 - GET /grades/student/:id/moyenne

Sécurité



Tous les endpoints protégés par JWT

Authorization: Bearer <token>



Tests des services REST via Postman

Auth Service

HTTP <http://localhost:8081/auth/me>

GET <http://localhost:8081/auth/me>

Send

Headers (9)

Body

Scripts

Settings

Cookies

Headers

Key Value Description

Authorization Bearer eyJhbGciOiJIUzI1NiJ9eyJyJ...
Key Value Description

Body Cookies Headers (10) Test Results

200 OK 1.61 s 380 B

{ } JSON ▾ Preview Visualize

```
1 {  
2   "role": "ADMIN",  
3   "id": 1,  
4   "email": "moe@example.com",  
5   "username": "moetez"  
6 }
```

HTTP <http://localhost:8081/auth/register>

POST <http://localhost:8081/auth/register>

Send

Docs Params Authorization Headers (8)

Body

Scripts Settings

Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Beautify

Body Cookies Headers (10) Test Results

200 OK 1.63 s 522 B

{ } JSON ▾ Preview Visualize

```
1 {  
2   "username": "moetez",  
3   "password": "1234",  
4   "email": "moe@example.com",  
5   "role": "ADMIN"  
6 }  
7
```

Body Cookies Headers (10) Test Results

200 OK 1.63 s 522 B

{ } JSON ▾ Preview Visualize

```
1 {  
2   "message": "User registered successfully",  
3   "accessToken": "eyJhbGciOiJIUzI1NiJ9.  
eyJyJ...  
j5nMm0zMZiphYnDjXQEDKbBDKgfC_avyYBv3yPfiJ9Y"  
4 }
```

HTTP <http://localhost:8081/auth/signin>

POST <http://localhost:8081/auth/signin>

Send

Docs Params Authorization Headers (8)

Body

Scripts Settings

Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Beautify

Body Cookies Headers (10) Test Results

200 OK 789 ms 512 B

{ } JSON ▾ Preview Visualize

```
1 {  
2   "username": "moetez",  
3   "password": "1234"  
4 }
```

Body Cookies Headers (10) Test Results

200 OK 789 ms 512 B

{ } JSON ▾ Preview Visualize

```
1 {  
2   "role": "ADMIN",  
3   "accessToken": "eyJhbGciOiJIUzI1NiJ9.  
eyJyJ...  
v6PuDwv0OCikJovJzwNqbHfYfe12SPuXanM_mijUps",  
4   "type": "Bearer"  
5 }
```

Tests des services REST via Postman

Grade Service

The screenshot shows three separate Postman requests:

- POST http://localhost:3002/grades**: A raw JSON payload is sent:

```
1 {
2   "studentId": "1",
3   "courseId": "INF101",
4   "noteDS": 12,
5   "noteExam": 14
6 }
```
- GET http://localhost:3002/grades/student/1**: Returns a single document:

```
1 [
2   {
3     "_id": "6927641b4bdcc828036e63ce3",
4     "studentId": "1",
5     "courseId": "INF100",
6     "noteDS": 12,
7     "noteExam": 14,
8     "moyenne": 13.20000000000001,
9     "etat": "ACQUIS",
10    "_v": 0
11  }
12 ]
```
- GET http://localhost:3002/grades/student/1/moyenne**: Returns a single document:

```
1 {
2   "studentId": "1",
3   "moyenne": 13.200000000000001,
4   "etat": "ACQUIS"
5 }
```

The screenshot shows a single Postman request:

GET http://localhost:3002/grades/student/1/moyenne

Body (8) Headers (8) Test Results (1)

200 OK 74 ms 570 B

Body (8) Cookies Headers (8) Test Results (1)

{} JSON ▾ Preview Visualize

```
1 [
2   {
3     "_id": "6927641b4bdcc828036e63ce3",
4     "studentId": "1",
5     "courseId": "INF100",
6     "noteDS": 12,
7     "noteExam": 14,
8     "moyenne": 13.20000000000001,
9     "etat": "ACQUIS",
10    "_v": 0
11  }
12 ]
```

Tests des services REST via Postman

Student Service

HTTP <http://localhost:3001/students>

POST <http://localhost:3001/students>

Body raw binary GraphQL JSON

```
1 {  
2   "firstName": "Saidane",  
3   "lastName": "Lamine",  
4   "email": "Lamine@example.com",  
5   "filiere": "Informatique",  
6   "niveau": "L3",  
7   "groupe": "B"  
8 }  
9
```

201 Created • 12 ms • 387 B • [Headers \(7\)](#) | [Test Results](#)

Body Cookies Headers (7) Test Results

{ } JSON ▾ ▶ Preview Visualize ▾

```
1 {  
2   "id": 1,  
3   "firstName": "Saidane",  
4   "lastName": "Lamine",  
5   "email": "Lamine@example.com",  
6   "filiere": "Informatique",  
7   "niveau": "L3",  
8   "groupe": "B",  
9   "matricule": ""  
10 }
```

HTTP <http://localhost:3001/students>

GET <http://localhost:3001/students>

Body raw binary GraphQL JSON

```
1 {  
2   "firstName": "Saidane",  
3   "lastName": "Lamine",  
4   "email": "Lamine@example.com",  
5   "filiere": "Informatique",  
6   "niveau": "L3",  
7   "groupe": "B"  
8 }  
9
```

200 OK • 14 ms • 384 B • [Headers \(7\)](#) | [Test Results](#)

Body Cookies Headers (7) Test Results

{ } JSON ▾ ▶ Preview Visualize ▾

```
1 [  
2   {  
3     "id": 1,  
4     "firstName": "Saidane",  
5     "lastName": "Lamine",  
6     "email": "Lamine@example.com",  
7     "filiere": "Informatique",  
8     "niveau": "L3",  
9     "groupe": "B",  
10    "matricule": ""  
11   }  
12 ]
```

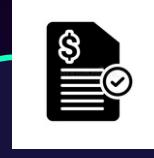
Services SOAP

Course Service



- Tech:
 - Spring Boot + SOAP/WSDL
- Endpoints:
 - addCourse(Course)
 - updateCourse(Course)
 - getCourseById(id)
 - listCourses()
 - listCourses()

Billing Service



- Tech:
 - Spring Boot + SOAP/WSDL
 - Endpoints:
 - calculateFees(studentId)
 - getInvoice(studentId)
 - payInvoice(invoiceId)
- Calcul: Total = Inscription + Scolarité + Pénalités

URL WSDL

`http://host:port/ws/courses?wsdl`
`http://host:port/ws/billing?wsdl`



Tests des services SOAP via Postman

Course Service



The image displays four screenshots of the Postman application interface, illustrating the testing of a Course Service API. The screenshots show various SOAP requests and their responses.

- Screenshot 1:** A POST request to `http://localhost:8082/ws` with raw XML body:

```
<soapenv:Header>
<soapenv:Body>
  <cou:addCourseRequest>
    <cou:code>CS101</cou:code>
    <cou:title>Introduction to CS</cou:title>
    <cou:teacherId>10</cou:teacherId>
```

Response status: 200 OK, 224 ms, 722 B. Body content (XML):

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
  <ns2:addCourseResponse xmlns:ns2="http://university.com/courses">
    <ns2:course>
      <ns2:id>2</ns2:id>
      <ns2:code>CS101</ns2:code>
      <ns2:title>Introduction to CS</ns2:title>
      <ns2:teacherId>10</ns2:teacherId>
      <ns2:niveau>L1</ns2:niveau>
      <ns2:groupe>B</ns2:groupe>
      <ns2:horaire>Monday 10:00</ns2:horaire>
    </ns2:course>
  </ns2:addCourseResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
- Screenshot 2:** A POST request to `http://localhost:8082/ws` with raw XML body:

```
<soapenv:Header/>
<soapenv:Body>
  <cou:addCourseRequest>
    <cou:course>
      <cou:id>1</cou:id>
      <cou:code>CS101</cou:code>
      <cou:title>Intro to CS</cou:title>
      <cou:teacherId>10</cou:teacherId>
      <cou:niveau>L1</cou:niveau>
      <cou:groupe>B</cou:groupe>
      <cou:horaire>Monday 10:00</cou:horaire>
    </cou:course>
  </cou:addCourseRequest>
</soapenv:Body>
```

Response status: 200 OK, 283 ms, 530 B. Body content (XML):

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
  <ns2:addCourseResponse xmlns:ns2="http://university.com/courses">
    <ns2:course>
      <ns2:id>1</ns2:id>
    </ns2:course>
  </ns2:addCourseResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
- Screenshot 3:** A POST request to `http://localhost:8082/ws` with raw XML body:

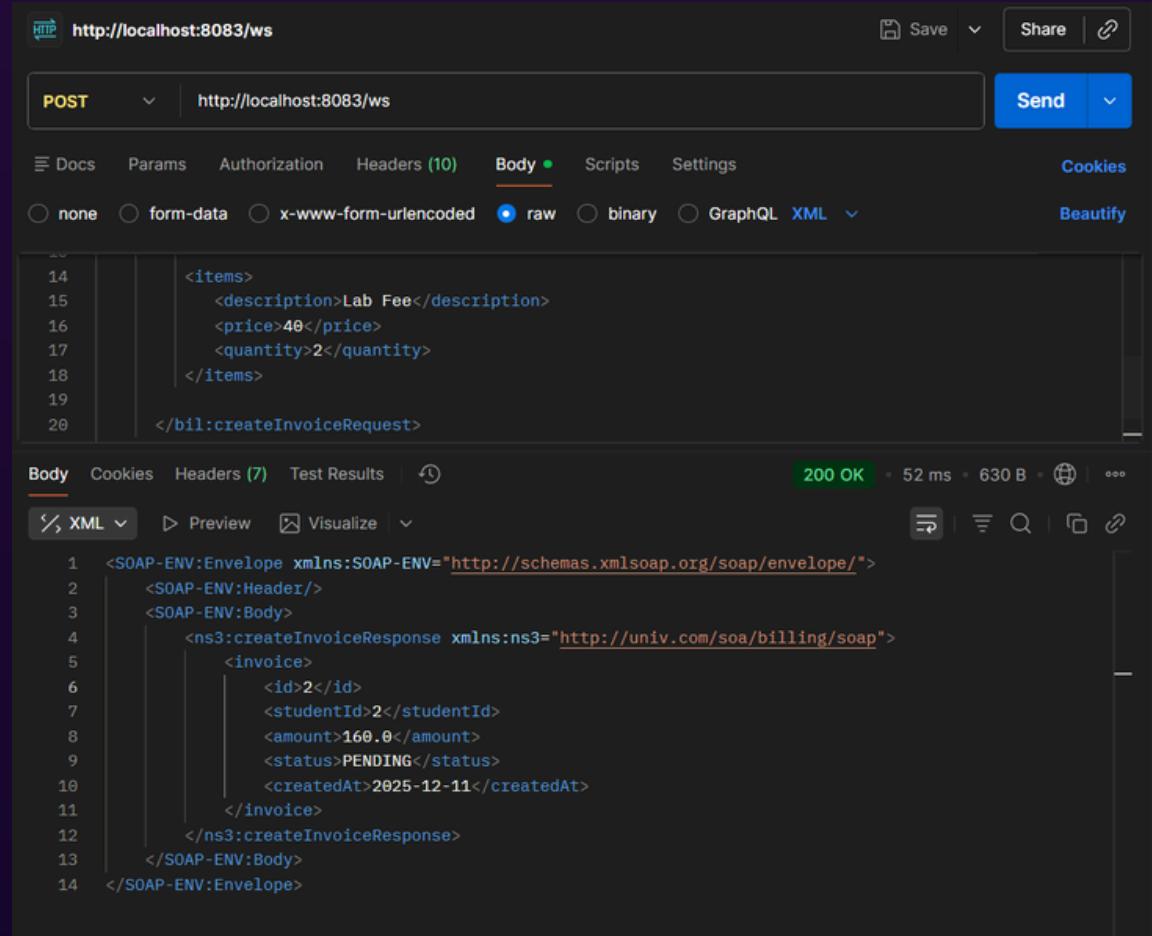
```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope">
<soapenv:Header/>
<soapenv:Body>
  <cou:getCourseRequest>
    <cou:id>1</cou:id>
  </cou:getCourseRequest>
</soapenv:Body>
</soapenv:Envelope>
```

Response status: 200 OK, 1.22 s, 464 B. Body content (XML):

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
  <ns2:getCourseResponse xmlns:ns2="http://university.com/courses">
    <ns2:course>
      <ns2:id>1</ns2:id>
    </ns2:course>
  </ns2:getCourseResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Tests des services SOAP via Postman

Billing Service

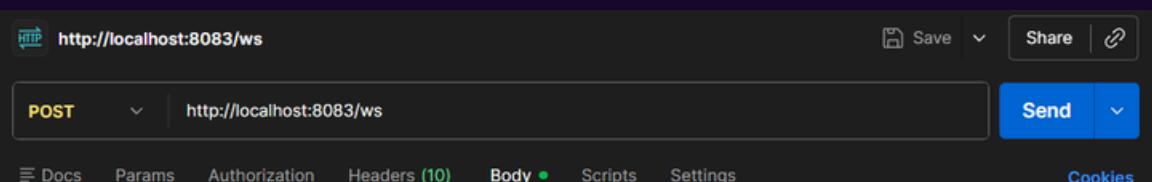


POST http://localhost:8083/ws

Body (raw XML)

```
<ns3:createInvoiceRequest xmlns:ns3="http://univ.com/soa/billing/soap">
  <items>
    <item>
      <description>Lab Fee</description>
      <price>40</price>
      <quantity>2</quantity>
    </item>
  </items>
</ns3:createInvoiceRequest>
```

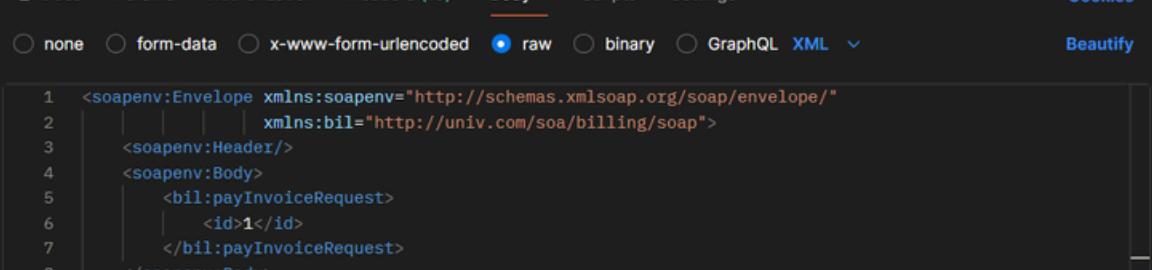
200 OK



POST http://localhost:8083/ws

Body (raw XML)

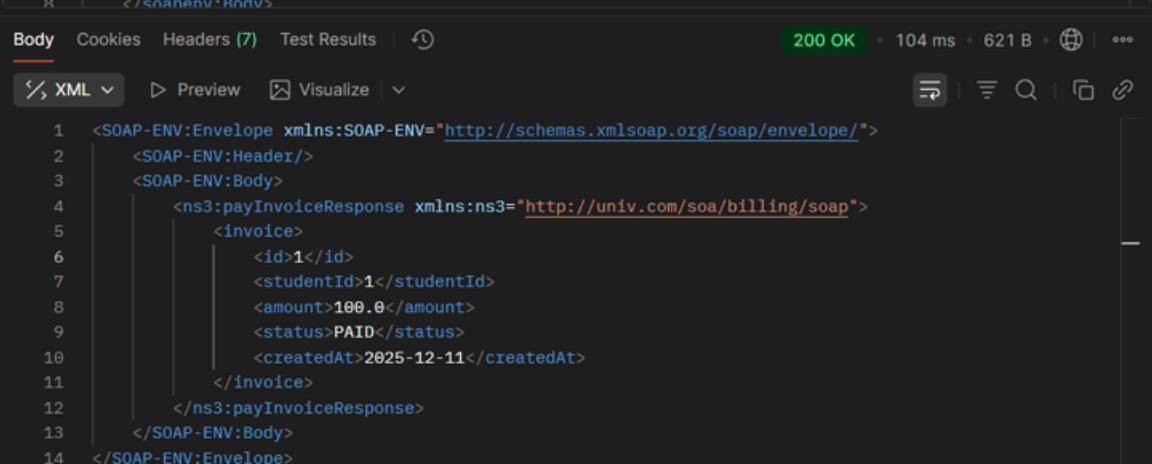
```
<ns3:createInvoiceResponse xmlns:ns3="http://univ.com/soa/billing/soap">
  <invoice>
    <id>2</id>
    <studentId>2</studentId>
    <amount>160.0</amount>
    <status>PENDING</status>
    <createdAt>2025-12-11</createdAt>
  </invoice>
</ns3:createInvoiceResponse>
```



POST http://localhost:8083/ws

Body (raw XML)

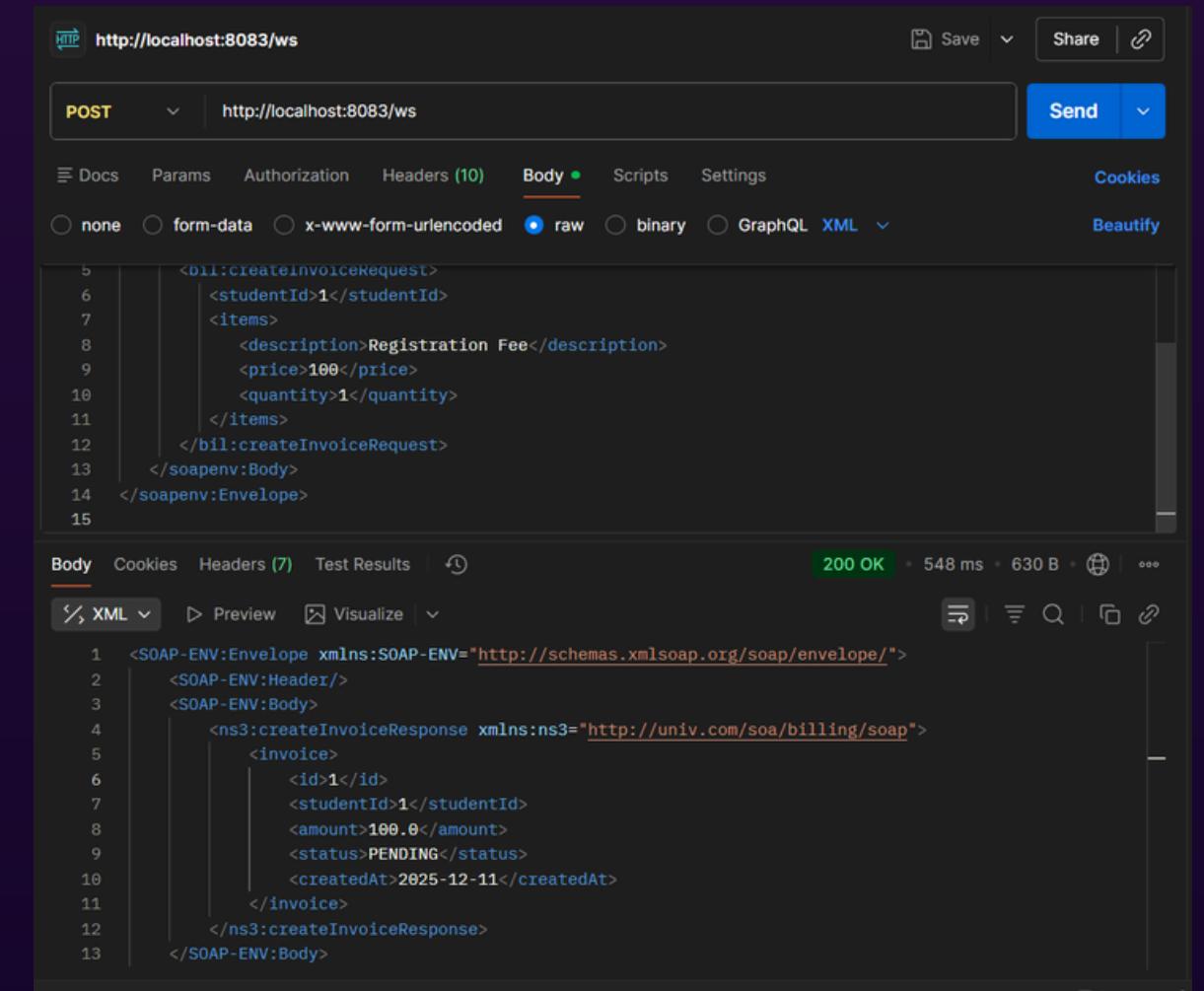
```
<ns3:payInvoiceRequest xmlns:ns3="http://univ.com/soa/billing/soap">
  <id>1</id>
</ns3:payInvoiceRequest>
```



POST http://localhost:8083/ws

Body (raw XML)

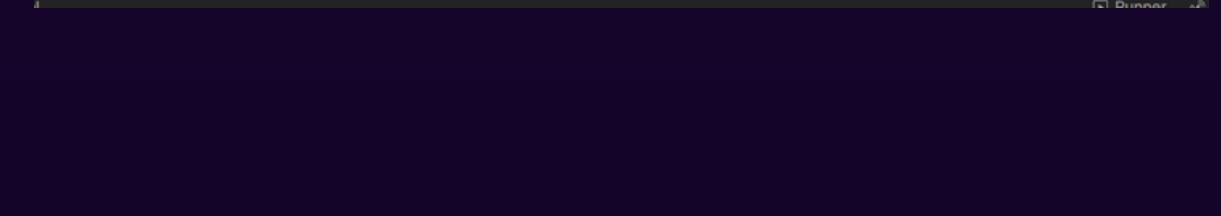
```
<ns3:payInvoiceResponse xmlns:ns3="http://univ.com/soa/billing/soap">
  <invoice>
    <id>1</id>
    <studentId>1</studentId>
    <amount>100.0</amount>
    <status>PAID</status>
    <createdAt>2025-12-11</createdAt>
  </invoice>
</ns3:payInvoiceResponse>
```



POST http://localhost:8083/ws

Body (raw XML)

```
<ns3:createInvoiceRequest xmlns:ns3="http://univ.com/soa/billing/soap">
  <studentId>1</studentId>
  <items>
    <item>
      <description>Registration Fee</description>
      <price>100</price>
      <quantity>1</quantity>
    </item>
  </items>
</ns3:createInvoiceRequest>
```



POST http://localhost:8083/ws

Body (raw XML)

```
<ns3:createInvoiceResponse xmlns:ns3="http://univ.com/soa/billing/soap">
  <invoice>
    <id>1</id>
    <studentId>1</studentId>
    <amount>100.0</amount>
    <status>PENDING</status>
    <createdAt>2025-12-11</createdAt>
  </invoice>
</ns3:createInvoiceResponse>
```



POST http://localhost:8083/ws

Body (raw XML)

```
<ns3:payInvoiceRequest xmlns:ns3="http://univ.com/soa/billing/soap">
  <id>1</id>
</ns3:payInvoiceRequest>
```

Périmètre Fonctionnel

Gestion Utilisateurs

- Création de comptes
- Authentification JWT
- Gestion des rôles

Gestion Étudiants

- CRUD complet
- Filtrage par filière/niveau

Gestion Notes

- Saisie des notes
- Calcul de moyennes
- État: PRINCIPALE/CONTROLE

Facturation

- Calcul des frais
- Gestion paiements
- Génération reçus



Périmètre Fonctionnel

Gestion Cours



- Crédit et modification
- Emploi du temps

API Gateway



- Point d'entrée unique
- Routage intelligent



Stack Technique



Spring Boot



Node.js

Express.js



JWT



SOAP/WSDL



REST API



Docker



Spring Cloud
Gateway



MongoDB



Stack Technique

Commandes de Démarrage

```
# Auth Service  
cd services/auth-service  
mvn spring-boot:run
```

↔

```
# Student Service  
cd services/student-service  
npm start
```

↔

```
# Grade Service  
cd services/grade-service  
npm start
```

```
# API Gateway  
cd services/api-gateway  
mvn spring-boot:run
```

↔

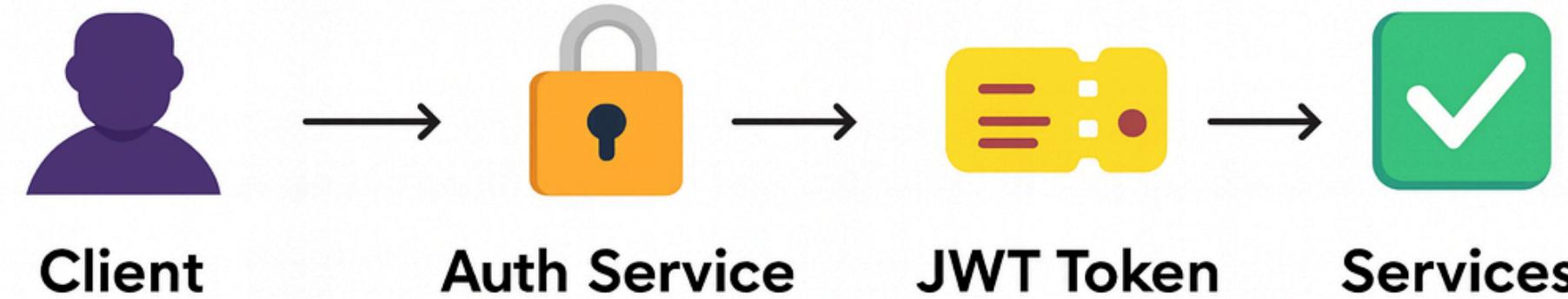
```
# Ou avec Docker Compose  
docker-compose up --build
```

Sécurité - JWT

Flux d'Authentification

🔒 JWT Security

Authentication Flow



Sécurité - JWT

Fonctionnalités

- Génération de token JWT
- Validation des tokens
- Gestion des rôles
- Passwords hashés (BCrypt)

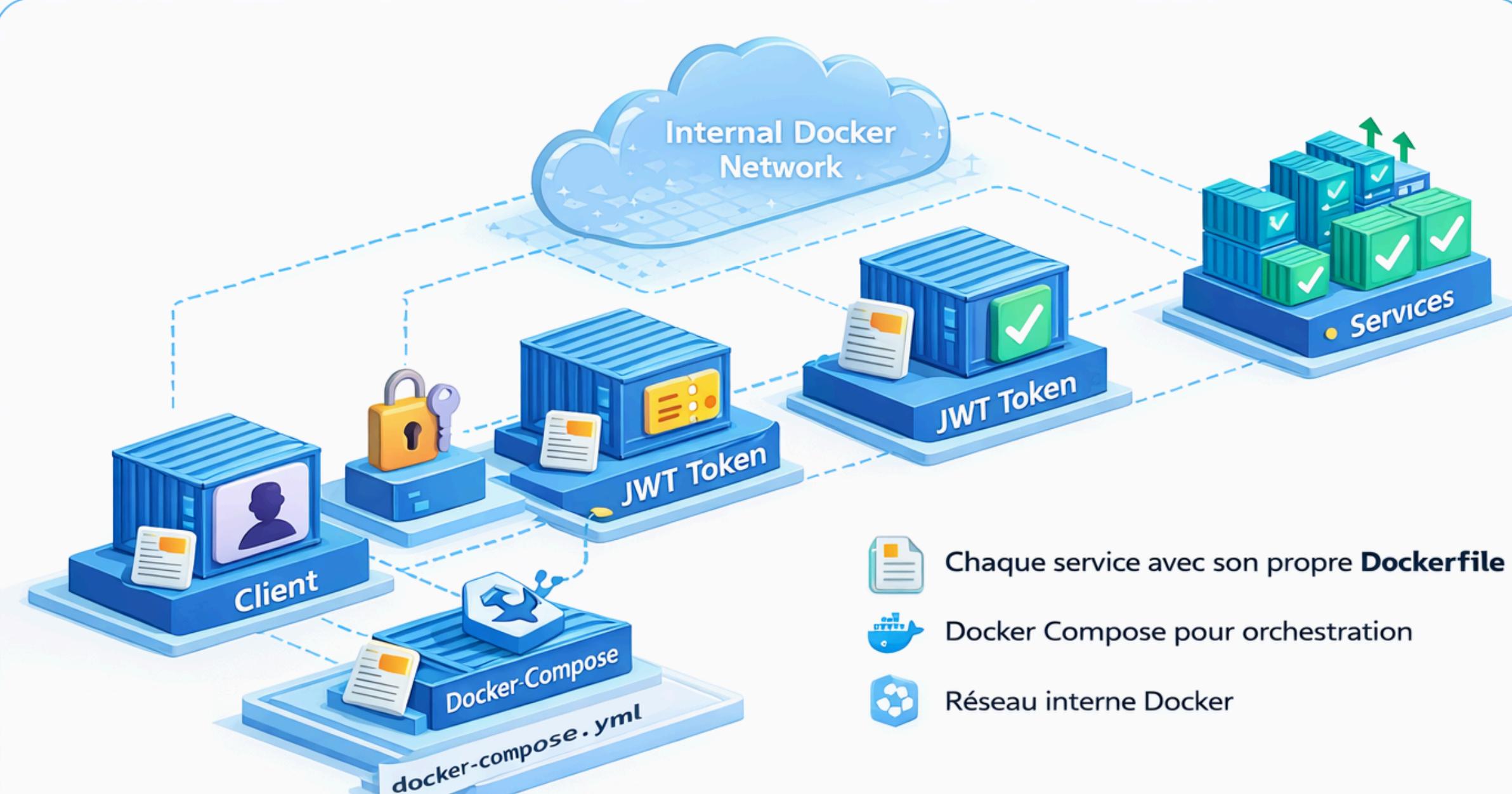
Rôles Utilisateurs

- ADMIN - Gestion complète
- TEACHER - Saisie notes
- STUDENT - Consultation

Déploiement Docker



Architecture Docker



Chaque service avec son propre **Dockerfile**



Docker Compose pour orchestration



Réseau interne Docker



Chaque service avec son propre **Dockerfile**
Docker Compose pour orchestration



Réseau interne Docker

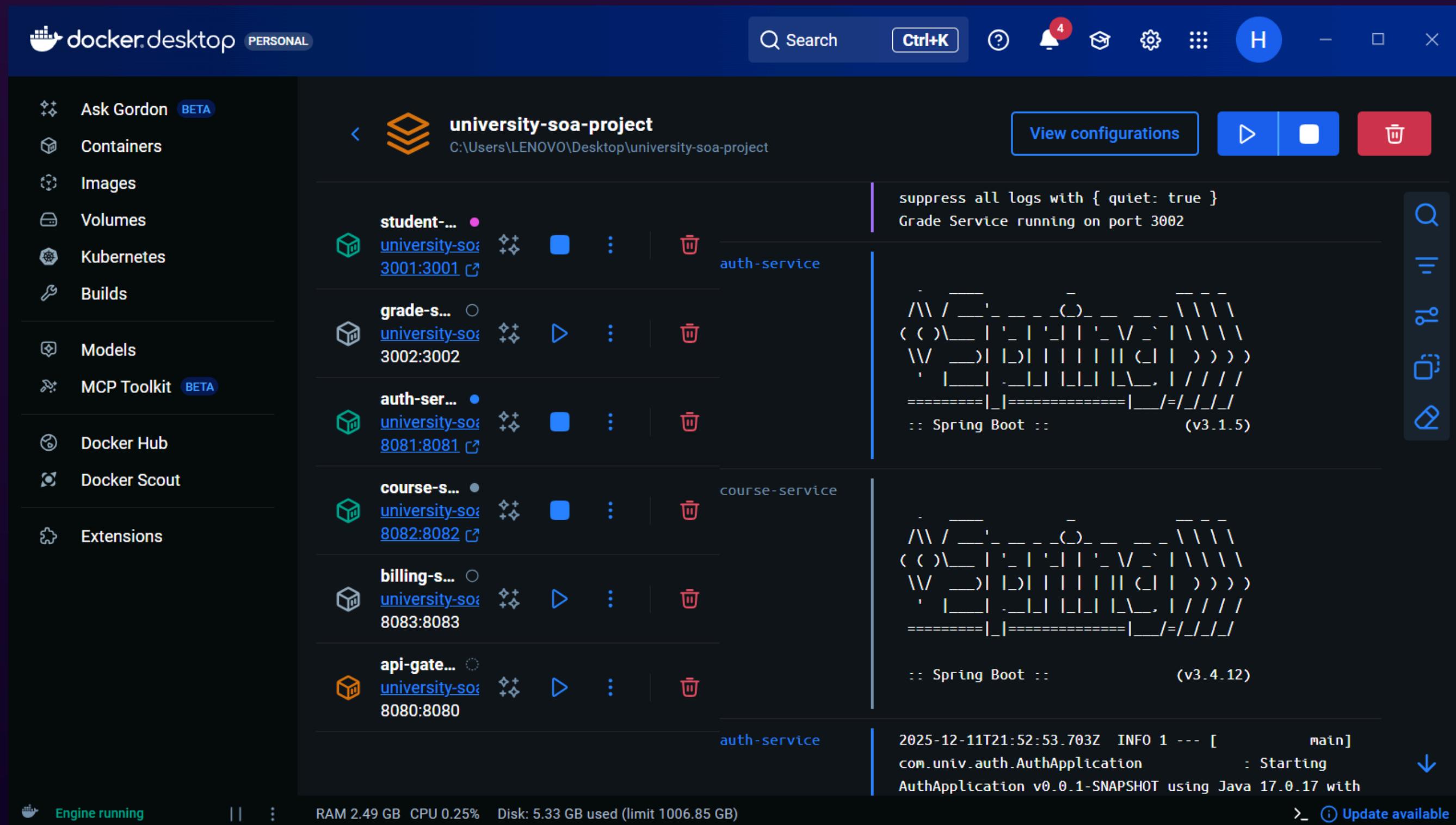
Déploiement Docker Dans Docker Desktop

The screenshot shows the Docker Desktop application window. The left sidebar has a 'Containers' tab selected. The main area displays a table of containers. One container is listed:

Name	Container ID	Image	Port(s)	CPU (%)	Actions
university-soa-p	-	-	-	1.39%	

At the bottom, status information includes 'Engine running', system resources ('RAM 2.44 GB CPU 0.25%', 'Disk: 5.33 GB used (limit 1006.85 GB)'), and an 'Update available' notification.

Déploiement Docker Dans Docker Desktop



Avantages de l'Architecture SOA

Évolutivité

- Services indépendants pouvant évoluer séparément sans impact sur le système global

Maintenabilité

- Code modulaire plus facile à maintenir, tester et déboguer

Interopérabilité

- Technologies hétérogènes (Java, Node.js) qui communiquent via standards

Scalabilité

- Chaque service peut être scalé indépendamment selon les besoins

Sécurité

- Authentification centralisée et protection par JWT de tous les endpoints

Déploiement

- Conteneurisation facilitant le déploiement et la gestion des environnements

Conclusion



- Ce projet a permis de réaliser une architecture SOA sécurisée et scalable, intégrant des services REST et SOAP avec une authentification basée sur JWT. L'interopérabilité entre Spring Boot et Node.js ainsi que le déploiement via Docker ont assuré flexibilité et portabilité.



Merci pour votre attention !

