

---

# Pilotage de projet DevOps

---

M2 MIAGE

Université Toulouse III - Paul Sabatier

Rémi Saurel

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Architecture</b>	<b>2</b>
2.1	Exercice VIII . . . . .	2
2.2	Exercice IX . . . . .	2
<b>3</b>	<b>Conclusion</b>	<b>3</b>
<b>4</b>	<b>Annexes</b>	<b>4</b>

# 1 Introduction

Ce projet de pilotage de projet DevOps a pour but de mettre en pratique les connaissances acquises lors du cours avec l'utilisation de Docker et la mise en place de plusieurs services. 2 exercices seront proposés dans ce rendu, à savoir les exercices VIII et IX. Toutes les justifications sont présentes dans la partie 4, mais vous pouvez retrouver toutes les sources dans **ce repository GitHub que j'ai créé**.

## 2 Architecture

### 2.1 Exercice VIII

#### Architecture générale

La figure 1 présente l'architecture générale de cet exercice. On y retrouve un fichier `docker-compose.yml` qui permet de lancer les 2 services : **app** et **db8**. Le service **app** est le projet Spring Boot fourni. Le service **db8** est une base de données MySQL.

#### Types de réseaux

Le réseau utilisé est le réseau **bridge** de Docker. Il permet de créer un réseau privé pour les conteneurs. Les ports 3306 et 8080 sont exposés pour pouvoir accéder à la base de données et à l'application Spring Boot.

#### Types de stockages

Le stockage utilisé est le stockage **volume** de Docker, ici nommé **db\_data**. Il permet de créer un volume pour les conteneurs. Il est utilisé pour la base de données afin de pouvoir conserver les données même si le conteneur est supprimé.

### 2.2 Exercice IX

#### Architecture générale

La figure 4 présente l'architecture générale de cet exercice. On y retrouve un fichier `docker-compose.yml` qui permet de lancer les 4 services : **app** et **db**. Le service **app** est le projet Spring Boot fourni. Le service **db** est une base de données MySQL. Le service phpMyAdmin est un service permettant d'interagir avec la base de données. Le service **Sonarcube** est un service permettant d'analyser le code source.

## Types de réseaux

Le réseau utilisé est le réseau **bridge** de Docker. Un réseau a été explicitement créé (le réseau 'exo9') dans le docker compose et permet aux différents services d'interagir entre eux. Les ports 3306, 8080 et 9000 sont exposés pour pouvoir accéder à la base de données, à l'application Spring Boot et à Sonarqube.

## Types de stockages

Comme pour l'exercice VIII, le stockage utilisé est le stockage **volume** de Docker, ici nommé **db\_data**. Il est utilisé pour la base de données afin de pouvoir conserver les données même si le conteneur est supprimé.

## 3 Conclusion

Ce projet m'a permis de mettre en pratique de façon concrète les éléments vus en cours. Cela m'a notamment servi afin de mieux comprendre les notions de volumes et de réseaux qui restaient assez floues. Pour conclure, cela m'a aidé à découvrir comment Docker pouvait être mis en place très facilement ainsi que les différents services qui pouvaient être utilisés.

## 4 Annexes

Vous pouvez retrouver toutes les sources sur le repository GitHub que j'ai créé.

### Exercice VIII

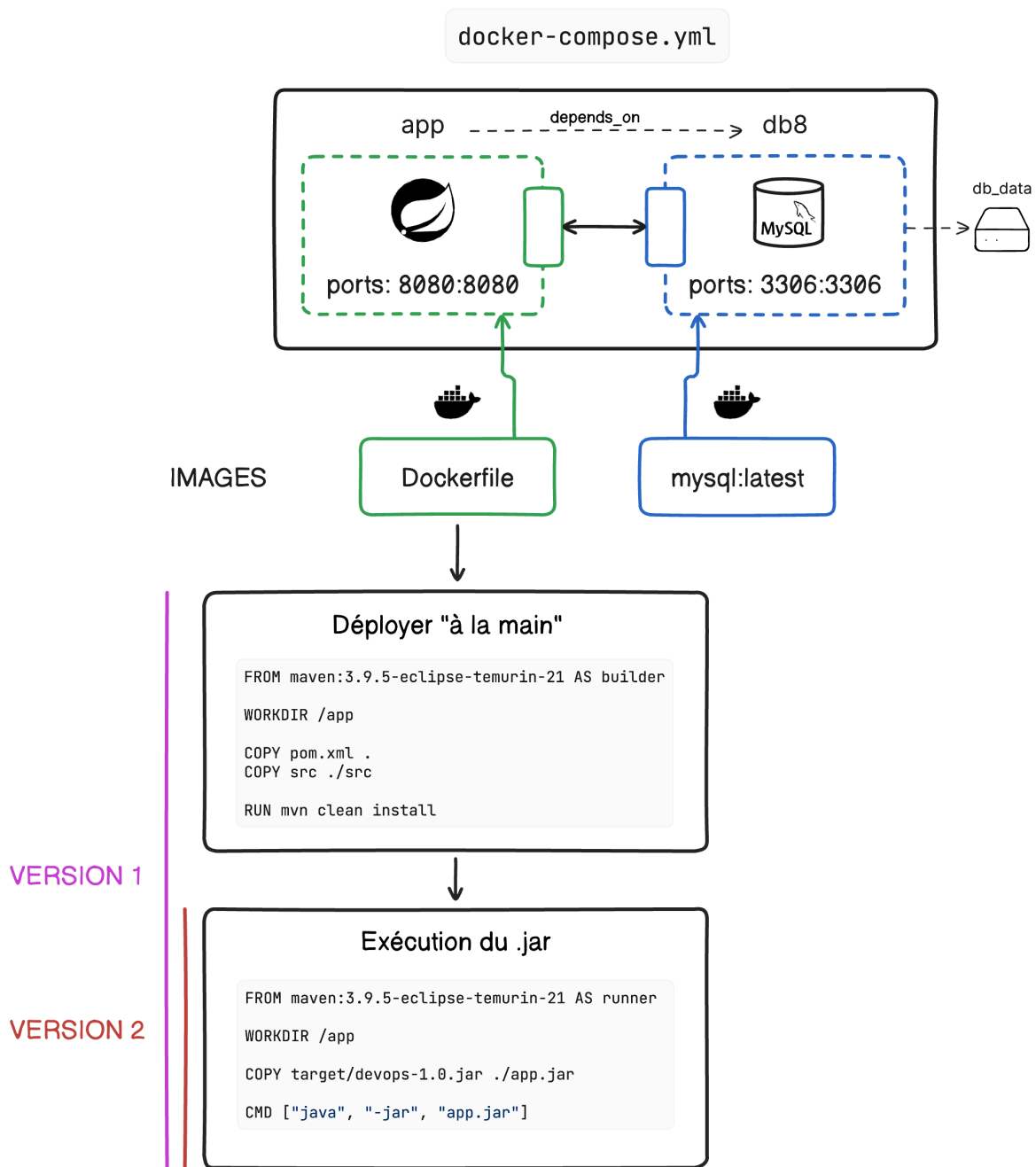


Figure 1: Architecture générale de l'exercice VIII

## Commandes

Commandes à réaliser : Dans le fichier Dockerfile, commenter / décommenter les lignes pour build ou non l'application Spring Boot.

```
docker compose up -d
```

Pour arrêter et supprimer les containers, vous pouvez utiliser :

```
docker compose down
```

## Résultats





exo8	Running (2/2)	0.97%	16 seconds ago
app-1 5048973789f6  <a href="#">exo8-app</a>	Running	0.29% <a href="#">8080:8080</a> 	16 seconds ago
db8 6178368b68ae  <a href="#">mysql:latest</a>	Running	0.68% <a href="#">3306:3306</a> 	16 seconds ago

Figure 2: Les services lancés sur Docker Desktop

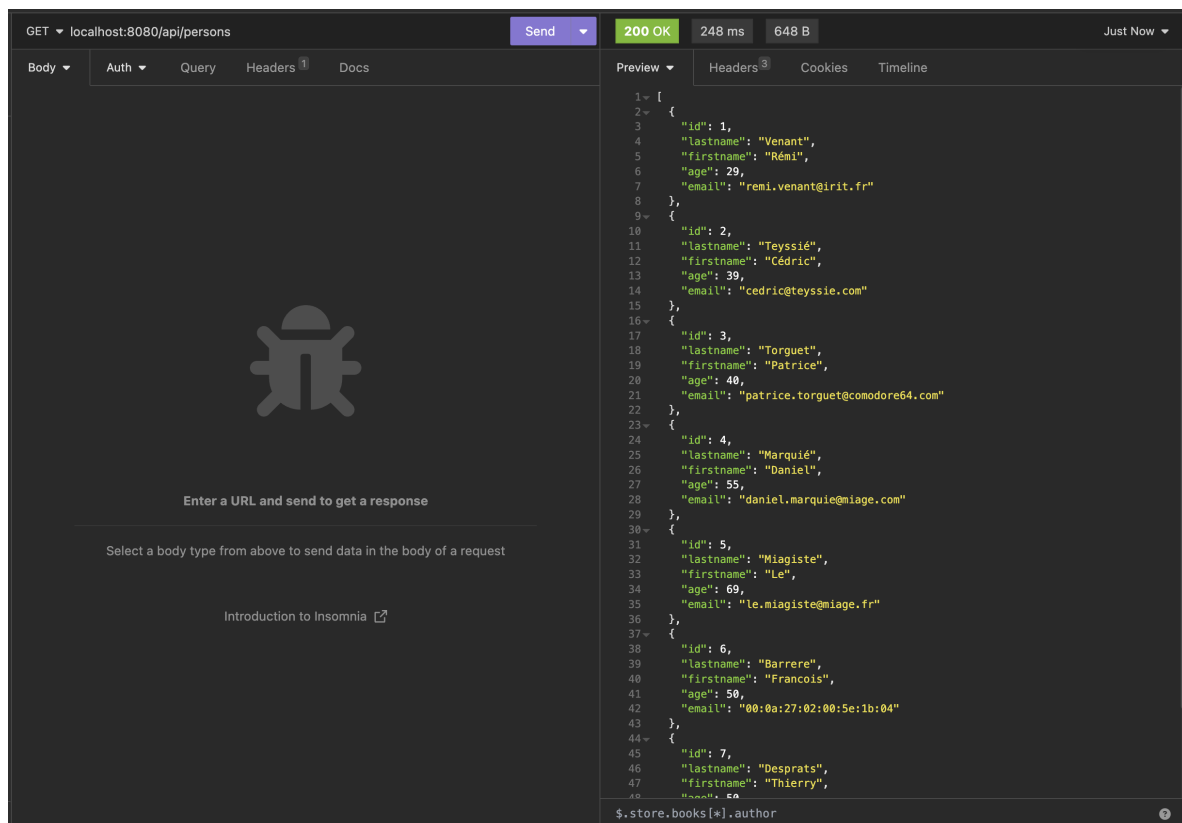


Figure 3: Test de l'app Spring Boot et de la BDD pour l'exercice 8

## Sources

Dockerfile :

```
# FIRST VERSION WITH BUILD
# FROM maven:3.9.5-eclipse-temurin-21 AS builder
#
# WORKDIR /app
#
# COPY pom.xml .
# COPY src ./src
#
# RUN mvn clean install
#
# FROM maven:3.9.5-eclipse-temurin-21 AS runner
#
# WORKDIR /app
#
# COPY --from=builder /app/target/*.jar ./app.jar
#
# ENTRYPOINT ["java", "-jar", "app.jar"]
```

```
# SECOND VERSION WITHOUT BUILD
FROM maven:3.9.5-eclipse-temurin-21 AS runner
```

```
WORKDIR /app
```

```
COPY target/devops-1.0.jar ./app.jar
```

```
CMD ["java", "-jar", "app.jar"]
```

---

docker-compose.yml :

```
version: '3'
```

```
services:
```

```
  db8:
```

```
    image: mysql:latest
```

```
    container_name: db8
```

```
    volumes:
```

```
      - db_data:/var/lib/mysql
```

```
      - ./data.sql:/docker-entrypoint-initdb.d/init.sql
```

```
    environment:
```

```
      MYSQL_USER: dev
```

```
      MYSQL_DATABASE: test
```

```
      MYSQL_ROOT_PASSWORD: root
```

```
      MYSQL_PASSWORD: root
```

```
    ports:
```

- "3306:3306"

app:

- build: .

- ports:

- "8080:8080"

- depends\_on:

- db8

volumes:

- db\_data:



## Exercice IX

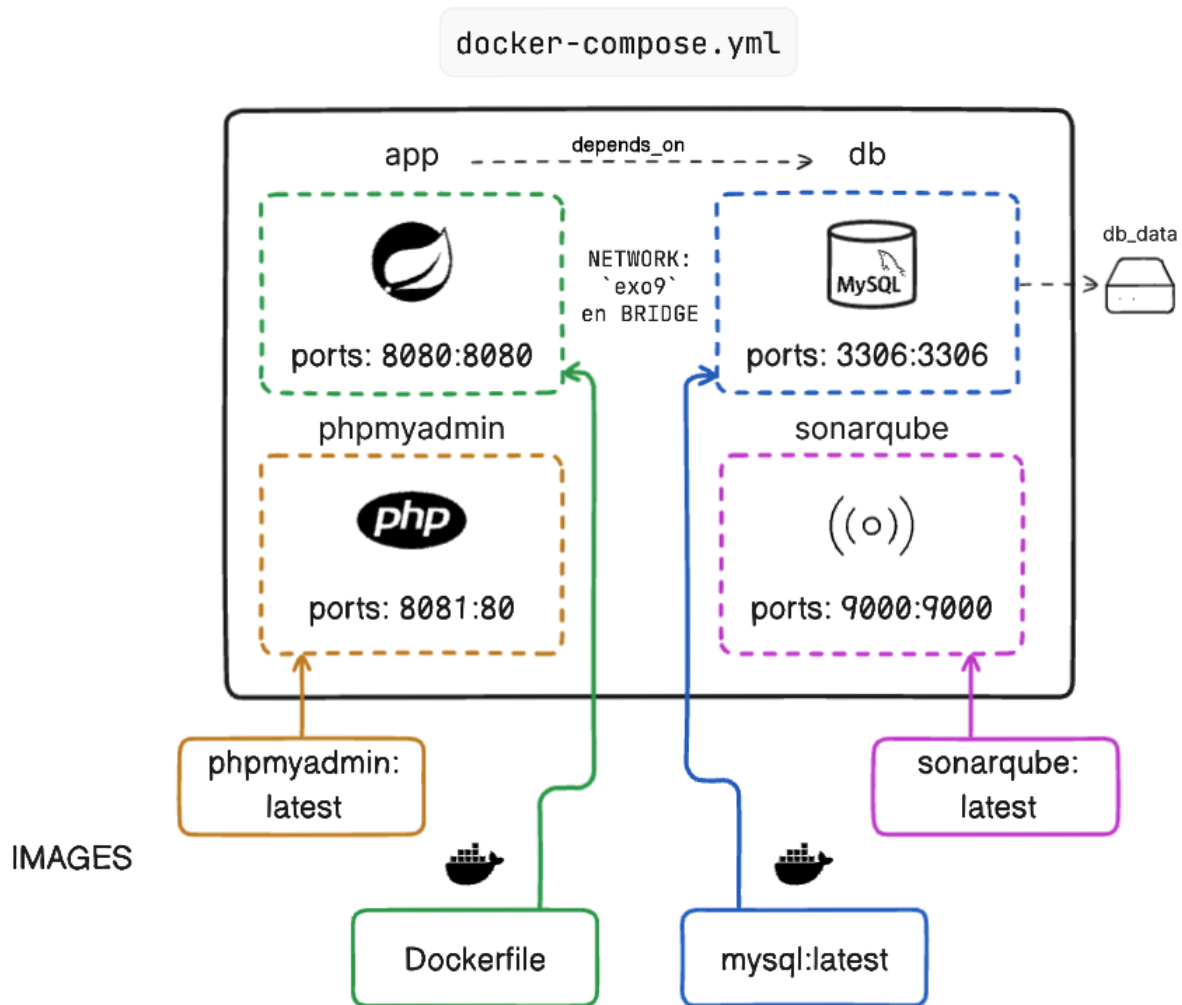


Figure 4: Architecture de l'exercice IX

### Commandes

Commandes à réaliser :

```
docker compose up -d
```

Pour arrêter et supprimer les containers, vous pouvez utiliser :

```
docker compose down
```

Vous pouvez retrouver les différents services sur les liens suivants :

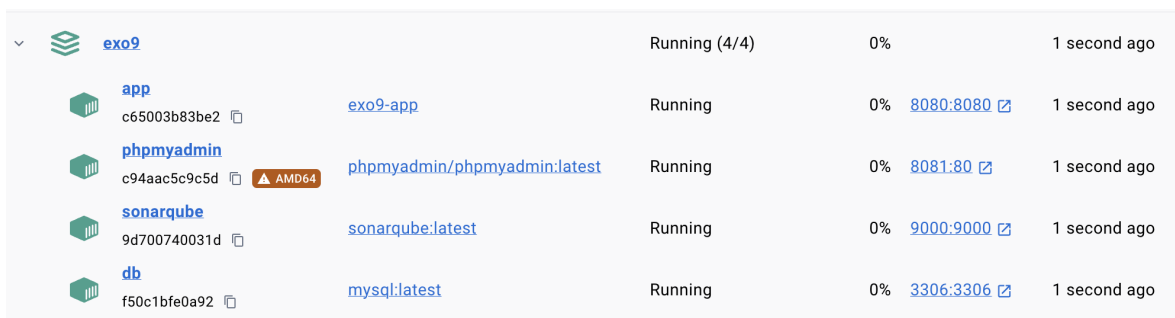
- <http://localhost:8080> pour l'application Spring Boot
- <http://localhost:8081> pour phpMyAdmin

- `http://localhost:9000` pour Sonarqube

Pour utiliser sonarqube, il faut créer un projet et générer un token. Ce token est à renseigner dans le projet Spring Boot si l'on souhaite que Sonarqube soit actualisé à chaque build. Dans mon cas, l'analyse peut être lancée grâce à la commande suivante :

```
mvn clean verify sonar:sonar \
  -Dsonar.projectKey=PROJECT_KEY \
  -Dsonar.projectName='PROJECT_NAME' \
  -Dsonar.host.url=http://localhost:9000 \
  -Dsonar.token=YOUR_TOKEN \
```

## Résultats



exo9		Running (4/4)	0%	1 second ago
<b>app</b>	exo9-app	Running	0% <a href="#">8080:8080</a>	1 second ago
<b>phpmyadmin</b>	phpmyadmin/phpmyadmin:latest	Running	0% <a href="#">8081:80</a>	1 second ago
<b>sonarqube</b>	sonarqube:latest	Running	0% <a href="#">9000:9000</a>	1 second ago
<b>db</b>	mysql:latest	Running	0% <a href="#">3306:3306</a>	1 second ago

Figure 5: Les services lancés sur Docker Desktop



Table	Action	Lignes	Type	Interclassement	Taille	Pe
<input type="checkbox"/> personnes	Parcourir Structure Rechercher Insérer Vider Supprimer	7	InnoDB	utf8mb3_general_ci	16,0 kio	
<input type="checkbox"/> personnes_seq	Parcourir Structure Rechercher Insérer Vider Supprimer	1	InnoDB	utf8mb4_0900_ai_ci	16,0 kio	
<b>2 tables</b>	<b>Somme</b>	<b>8</b>	<b>InnoDB</b>	<b>utf8mb4_0900_ai_ci</b>	<b>32,0 kio</b>	

Figure 6: Interface phpMyAdmin avec la table 'personnes'

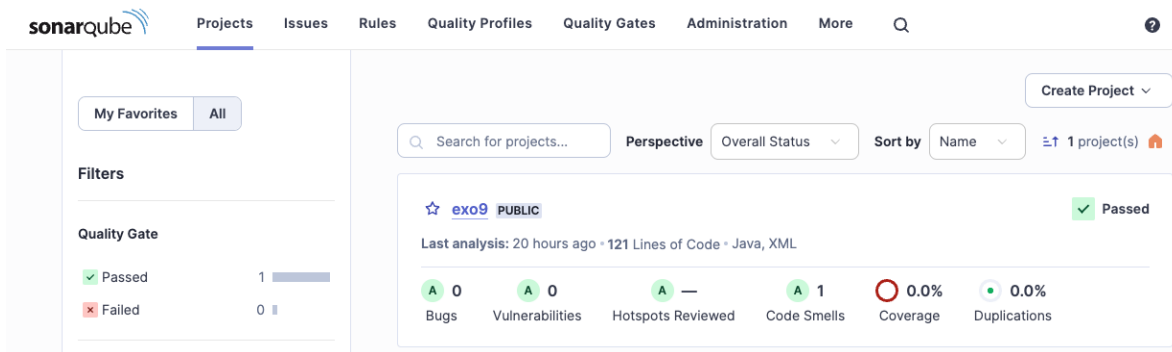


Figure 7: Interface Sonarqube

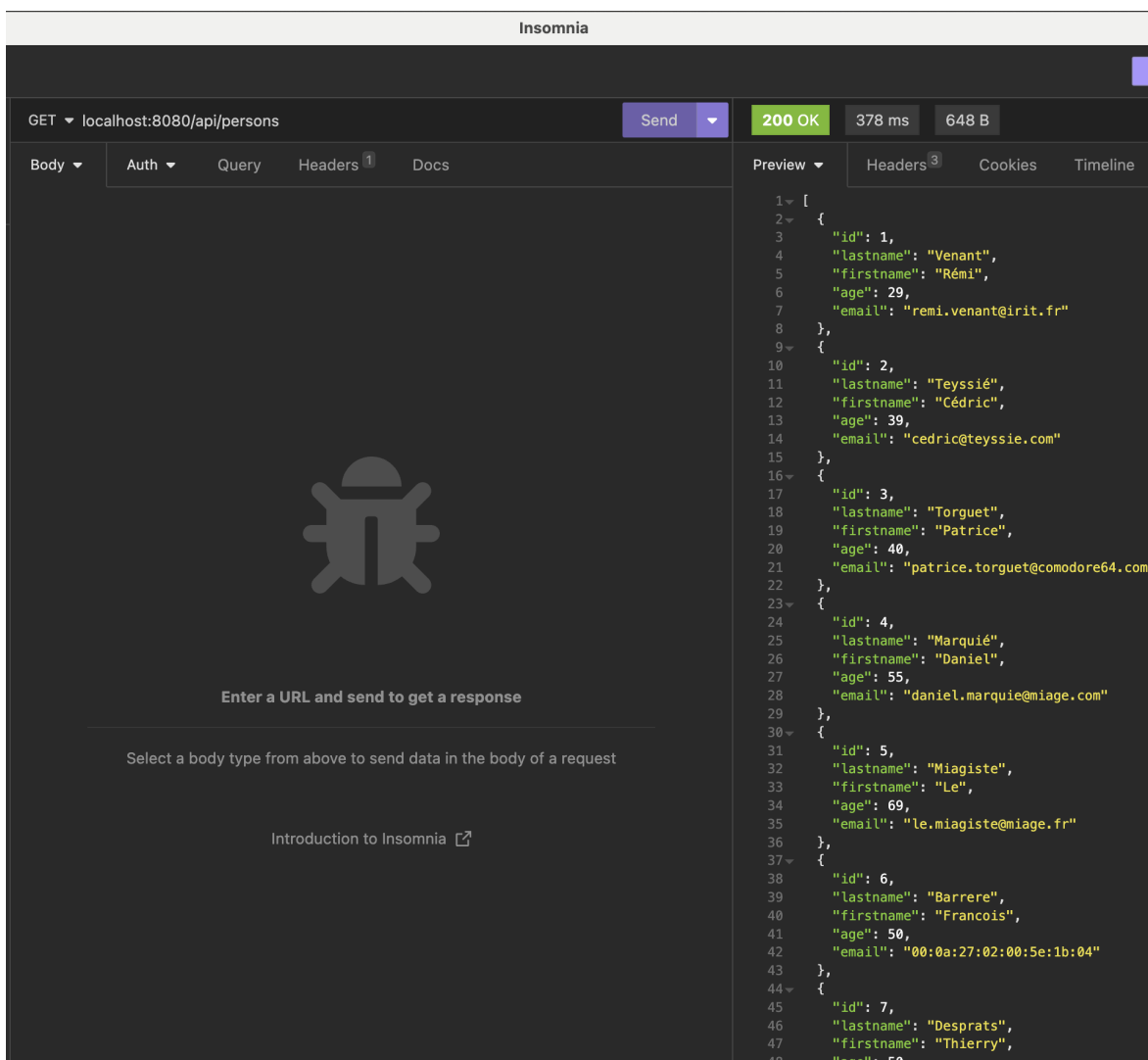


Figure 8: Test de l'app Spring Boot et de la BDD pour l'exercice 9

## Sources

Dockerfile :

```
FROM maven:3.9.5-eclipse-temurin-21 AS builder
```

```
WORKDIR /app
```

```
COPY target/devops-1.0.jar ./app.jar
```

```
CMD ["java", "-jar", "app.jar"]
```

---

docker-compose.yml :

```
version: '3'
```

```
services:
```

```
  db:
```

```
    image: mysql:latest
```

```
    container_name: db
```

```
    volumes:
```

```
      - db_data:/var/lib/mysql
```

```
      - ./data.sql:/docker-entrypoint-initdb.d/init.sql
```

```
    environment:
```

```
      MYSQL_USER: dev
```

```
      MYSQL_DATABASE: test
```

```
      MYSQL_ROOT_PASSWORD: root
```

```
      MYSQL_PASSWORD: root
```

```
    ports:
```

```
      - "3306:3306"
```

```
    networks:
```

```
      - exo9
```

```
  app:
```

```
    build: .
```

```
    container_name: app
```

```
    ports:
```

```
      - "8080:8080"
```

```
    depends_on:
```

```
      - db
```

```
    networks:
```

```
      - exo9
```

```
  sonarqube:
```

```
    image: sonarqube:latest
```

```
    container_name: sonarqube
```

```
    ports:
```

```
      - "9000:9000"
```

```

environment:
  - SONARQUBE_JDBC_URL=jdbc:mysql://db:3306/sonarqube
    ?useUnicode=true
    &characterEncoding=utf8
    &rewriteBatchedStatements=true
    &useConfigs=maxPerformance
    &useSSL=false
  - SONARQUBE_JDBC_USERNAME=dev
  - SONARQUBE_JDBC_PASSWORD=root
networks:
  - exo9

phpmyadmin:
  image: phpmyadmin/phpmyadmin:latest
  container_name: phpmyadmin
  environment:
    - PMA_ARBITRARY=1
    - PMA_HOST=db
    - PMA_PORT=3306
  ports:
    - "8081:80"
  networks:
    - exo9

networks:
  exo9:
    driver: bridge

volumes:
  db_data:

```