

EMSI

PROJET : ARCHITECTURE DES
COMPOSANTS D'ENTREPRISE

ANAS ZAKARI/GROUPE 2

GESTION DES COURS SUPPLÉMENTAIRES

January 21, 2024

Introduction

- Aperçu du projet

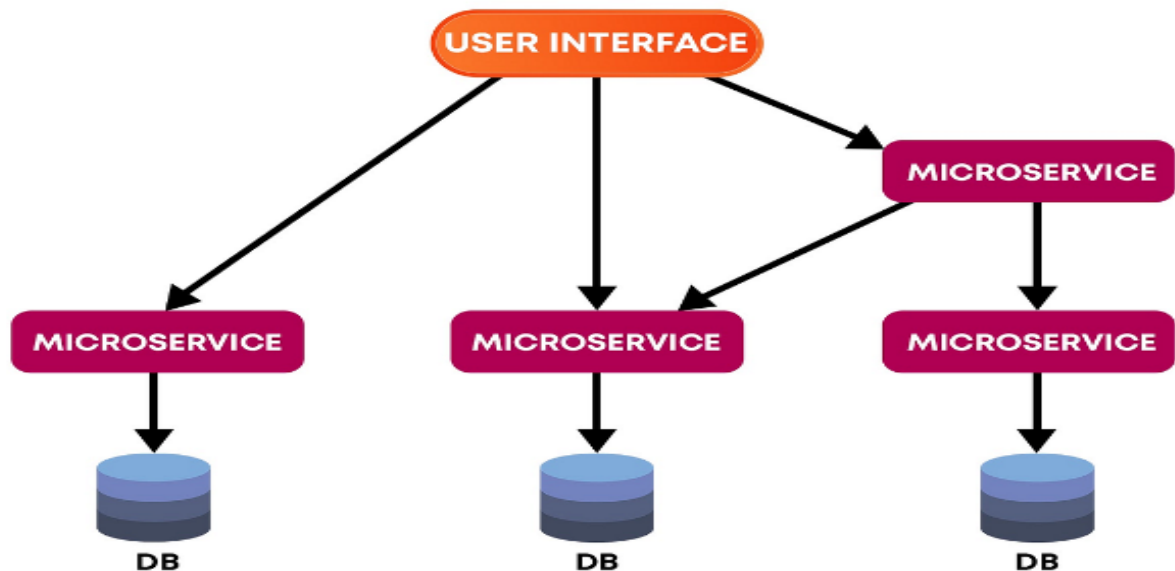
Ce projet vise à gérer les heures supplémentaires pour une école. On peut gérer les professeurs et leurs disponibilités par rapport au temps, ainsi que les groupes avec leurs étudiants. Le projet permet également de gérer les étudiants en créant pour chacun une carte basée sur leurs informations personnelles. De plus, il prend en charge la gestion des absences, des paiements, des emplois du temps, et enfin, des salles.

- Importance de l'architecture microservices

1. Les microservices favorisent l'évolutivité et l'agilité dans le développement logiciel. Chaque microservice peut être développé, déployé et développé de manière autonome, ce qui permet aux équipes de travailler de manière agile et de répondre rapidement aux changements.
2. Découplage : Les microservices sont conçus pour fonctionner séparément, ce qui favorise le découplage. Cela signifie que si un microservice échoue, cela n'affectera pas nécessairement l'ensemble du système, ce qui augmente la robustesse et la tolérance aux erreurs.
3. Isolation des responsabilités : chaque microservice peut se concentrer sur une tâche ou une fonctionnalité particulière, ce qui simplifie le développement, la maintenance et la compréhension du code. Chaque service est responsable d'une portion de la logique commerciale.
4. Le remplacement ou la mise à niveau d'un microservice particulier est plus simple sans affecter le reste du système, ce qui simplifie la gestion du cycle de vie de l'application.

- Architecture Microservices

GATEWAY
SERVICE-ABSENCE
SERVICE-COURS
SERVICE-EMPLOIDUTEMPS
SERVICE-ETUDIANT
SERVICE-GROUPE
SERVICE-PAIEMENT
SERVICE-PROF
SERVICE-RESERVATION
SERVICE-SALLE

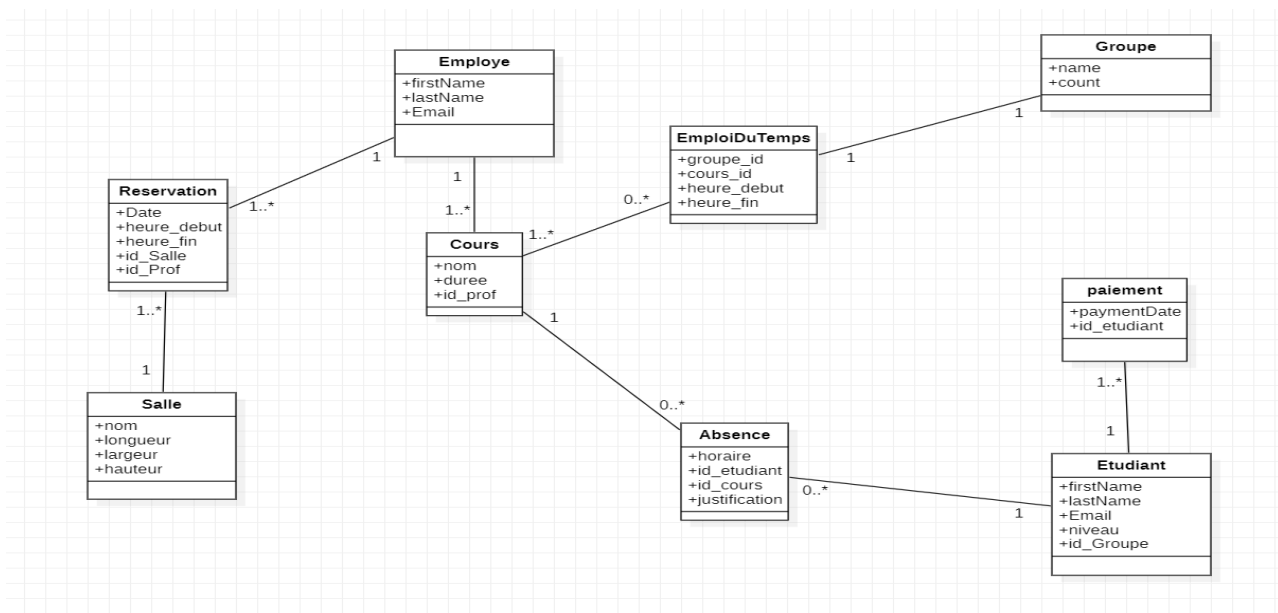


- Description des services

1. **Gateway** : C'est pour faciliter l'appel des API, par exemple <http://localhost:Port-Gateway/Nom-Service>.
2. **SERVICE-ABSENCE** : Pour gérer l'absence des étudiants, fait l'appel des microservices suivants : etudiant, groupe.
3. **SERVICE-COURS** : Pour gérer les cours en affectant à chaque cours un prof, fait l'appel du microservice PROF.
4. **SERVICE-EMPLOIDUTEMPS** : Pour gérer les emplois du temps pour les groupes et profs.

5. **SERVICE-ETUDIANT** : Pour gérer les étudiants en affectant à chaque étudiant un groupe, fait l'appel du microservice groupe.
6. **SERVICE-PAIEMENT** : Gérer les paiements des étudiants.
7. **SERVICE-PROF** : Gérer les professeurs.
8. **SERVICE-RESERVATION** : Gérer les réservations des salles par les profs dans un horaire spécifique.
9. **SERVICE-SALLE** : Gérer les salles avec leur matériel.

- Conception des Microservices



SERVICE-ETUDIANT

Responsabilités : Ce service est responsable de la gestion des informations des étudiants, notamment l'ajout, la mise à jour et la suppression des étudiants. Il doit également prendre en charge l'affectation de chaque étudiant à un groupe spécifique.

Interface/ API : Définissez les points d'entrée du service pour permettre l'ajout, la mise à jour et la suppression des étudiants. Par exemple :

POST /etudiants : Pour ajouter un nouvel étudiant.

PUT /etudiants/id : Pour mettre à jour les informations d'un étudiant existant.

DELETE /etudiants/id : Pour supprimer un étudiant.

GET /etudiants/id : Pour obtenir les détails d'un étudiant.

Affectation au Groupe : Intégrez la logique d'affectation des étudiants à un groupe spécifique. Cela peut impliquer des appels au microservice

”groupe” pour récupérer les détails des groupes et les affecter correctement.

SERVICE-RESERVATION

Responsabilités : Le service de réservation est responsable de la gestion des réservations de salles pour les professeurs, en veillant à éviter les conflits d’horaire et en assurant une allocation efficace des ressources.

Interface/API : Définissez les points d’entrée du service pour permettre la création, la mise à jour et la suppression des réservations. Par exemple :

POST /reservations : Pour créer une nouvelle réservation.

PUT /reservations/id : Pour mettre à jour les détails d’une réservation existante.

DELETE /reservations/id : Pour annuler une réservation.

GET /reservations/id : Pour obtenir les détails d’une réservation.

Gestion des Conflits : Intégrez une logique qui vérifie et évite les conflits d’horaire lors de la création ou de la modification d’une réservation. Cela peut impliquer la vérification de la disponibilité des salles et la consultation du calendrier des réservations existantes.

Affectation au Professeur : Si nécessaire, assurez-vous que chaque réservation est associée au professeur concerné. Cela peut impliquer des appels au microservice ”SERVICE-PROF” pour récupérer les détails du professeur.

SERVICE-ABSENCE

Responsabilités : Le service de gestion des absences est responsable de la gestion des

absences des étudiants. Il prend en charge la création d'incidents d'absence, l'attribution d'une raison d'absence, et il peut interagir avec les microservices "etudiant" et "groupe" pour obtenir des informations pertinentes.

Interface/API : Définissez les points d'entrée du service pour permettre la création, la mise à jour et la consultation des incidents d'absence. Par exemple :

POST /absences : Pour enregistrer une nouvelle absence.

PUT /absences/id : Pour mettre à jour les détails d'un incident d'absence existant.

GET /absences/id : Pour obtenir les détails d'un incident d'absence.

Affectation à l'Étudiant et au Groupe : Intégrez la logique pour lier chaque incident d'absence à un étudiant spécifique et à son groupe respectif. Cela peut impliquer des appels aux microservices "etudiant" et "groupe" pour récupérer les informations nécessaires.

SERVICE-emploiDuTemps

Responsabilités : Le service de gestion des emplois du temps est chargé de créer, mettre à jour et consulter les emplois du temps pour les groupes et les professeurs, en tenant compte des contraintes d'horaires et de disponibilité.

Interface/API : Définissez les points d'entrée du service pour permettre la création, la mise à jour et la consultation des emplois du temps. Par exemple :

POST /emploiutemps : Pour créer un nouvel emploi du temps.

PUT /emploiutemps/id : Pour mettre à jour les détails d'un emploi du temps existant.

GET /emploiutemps/id : Pour obtenir les détails d'un emploi du temps.

Affectation au Groupe et au Professeur : Intégrez la logique pour attribuer chaque emploi du temps à un groupe spécifique et à un professeur donné. Cela peut impliquer des appels aux microservices "groupe" et "prof" pour récupérer les informations nécessaires.

- Mécanismes de communication

La communication entre les microservices a été réalisée à travers le "Rest-Template".

"RestTemplate" est une classe fournie par Spring Framework, une infrastructure pour le développement d'applications Java, qui facilite la communication avec les services RESTful. Elle simplifie l'appel des services HTTP en encapsulant de nombreuses opérations courantes et en offrant une interface conviviale.

- Conteneurisation avec Docker

1. Création des Dockerfiles : Pour chaque microservice, un fichier Dockerfile est créé pour définir les dépendances, les étapes d'installation et la configuration nécessaire. .

```
# Specify the base image with Maven for building
FROM maven:3.8-openjdk-17 AS builder

# Set the working directory in the container
WORKDIR /app

# Copy the source code and pom.xml to the container
```



```
COPY ./src ./src
COPY ./pom.xml .

# Build the Maven project
RUN mvn clean package

# Specify the base image for the final image
FROM openjdk:17-jdk-alpine

# Create a volume for temporary files
VOLUME /tmp

# Define an argument for the JAR file
ARG JAR_FILE=target/*.jar

# Copy the compiled JAR file from the builder stage
  to the final image
COPY --from=builder /app/${JAR_FILE} app.jar

# Set the entry point for the container
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

2. Configuration du Docker Compose : Un fichier docker-compose.yml est créé pour définir les services, les dépendances et les configurations réseau..

```
    version: '3'

services:
  mysql:
    image: mysql:latest
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: HSupp
    ports:
      - "3306:3306"

  backend:
    build:
      context: ./JEEMicroServices
    ports:
      - "9090:9090"
    depends_on:
      - mysql
    environment:
      SPRING_DATASOURCE_URL: jdbc:mysql://mysql:3306/HSupp
      SPRING_DATASOURCE_USERNAME: root
      SPRING_DATASOURCE_PASSWORD: root
    healthcheck:
      test: "/usr/bin/mysql --user=root --password=root --
        execute \"SHOW DATABASES;\""
      interval: 5s
      timeout: 2s
      retries: 100

  frontend:
    build:
      context: ./JEEMicroServicesFront
```

```

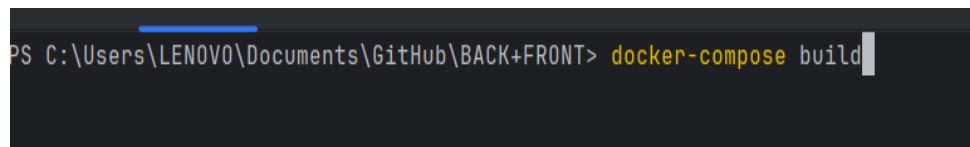
ports:
  - "91:91"

depends_on:
  - backend

phpmyadmin:
  image: phpmyadmin/phpmyadmin
  environment:
    PMA_HOST: mysql
    PMA_PORT: 3306
    MYSQL_ROOT_PASSWORD: root
  ports:
    - "8081:80"

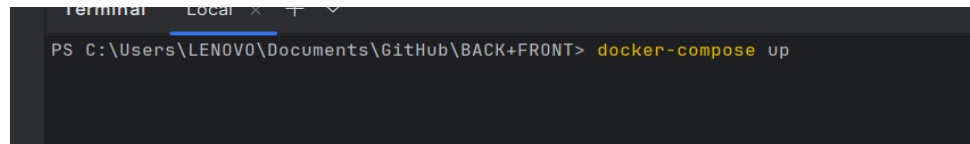
```

3. Execution de la commande



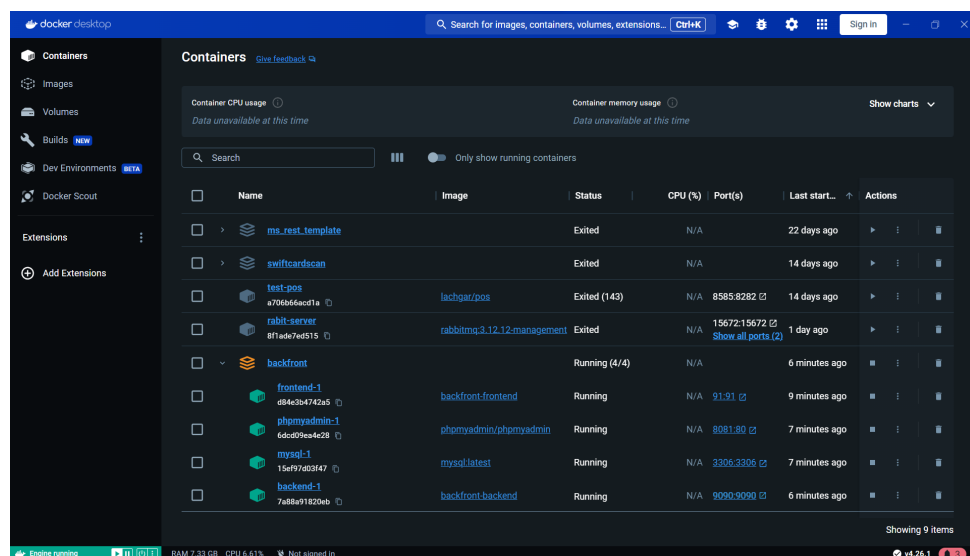
```
PS C:\Users\LENOVO\Documents\GitHub\BACK+FRONT> docker-compose build
```

4. Execution de la commande



```
PS C:\Users\LENOVO\Documents\GitHub\BACK+FRONT> docker-compose up
```

5. Resultat



Name	Image	Status	CPU (%)	Port(s)	Last start...	Actions
ms_rest_template		Exited	N/A		22 days ago	
swiftcardscan		Exited	N/A		14 days ago	
test-pos	lachgar/pos	Exited (143)	N/A	8585:8282	14 days ago	
rabit-server	rabbitmq:3.12.12-management	Exited	N/A	15672:15672	1 day ago	
backfront		Running (4/4)	N/A		6 minutes ago	
frontend-1	backfront-frontend	Running	N/A	91:91	9 minutes ago	
phpmyadmin-1	phpmyadmin/phpmyadmin	Running	N/A	8081:80	7 minutes ago	
mysql-1	mysql:latest	Running	N/A	3306:3306	7 minutes ago	
backend-1	backfront-backend	Running	N/A	9090:9090	6 minutes ago	

- CI/CD avec Jenkins

GitHub Repository

```
pipeline {
    agent any

    tools {
        maven 'maven'
    }

    stages {
        stage('Git_Clone') {
            steps {
                script {
                    checkout([$class: 'GitSCM',
                        branches: [[name: 'main']],
                        userRemoteConfigs: [[url: 'https://github.com/Zakari-
                        Anas/JENKINS-BACK-FRONT.git'
                        ]]])
                }
            }
        }

        stage('Build_Discovery-Service') {
            steps {
                dir('JEEMicroServices/EurekaServer1')
```

```

    ) {

        bat 'mvn_clean_install'
    }
}

stage('Build_gateway-service') {
    steps {
        dir('JEEMicroServices/GateWay') {

            bat 'mvn_clean_install'
        }
    }
}

stage('Build_PROF-service') {
    steps {
        script{
            dir('JEEMicroServices/microService-
                employe') {

                withSonarQubeEnv('sonarserver') {
                    bat "mvn_clean_verify_sonar:
                        sonar_-Dsonar.projectKey=
                        geocoder-service_-Dsonar."
                }
            }
        }
    }
}

```

```

        projectName='geocoder-service'
        "
    }

    timeout(time:1, unit:'HOURS'){
        def qg = waitForQualityGate()
        if (qg.status != 'OK') {
            error "Quality_Gate_did_
                    not_pass._Check_the_
                    SonarQube_dashboard_
                    for_details."
        }
    }

    bat 'mvn_clean_install'
    }
}

stage('Build_group-service') {
    steps {
        script{
            dir('JEEMicroServices/microService-
                groupe'){

                    withSonarQubeEnv('sonarserver')

```

```

        {
            bat "mvn_clean_verify_sonar:
                sonar-Dsonar.projectKey=
                weather-service-Dsonar.
                projectName='weather-
                service' "
        }
        timeout(time:1, unit:'HOURS'){
            def qg = waitForQualityGate
                ()
            if (qg.status != 'OK') {
                error "Quality_Gate_did_
                    not_pass._Check_the_
                    SonarQube_dashboard_
                    for_details."
            }
        }
    }

    bat 'mvn_clean_install'
    }}
}

stage('Build_paielement-service') {
    steps {
        script{
            dir('JEEMicroServices/microService-

```

```

        paiement') {

        withSonarQubeEnv('sonarserver') {
            bat "mvn_clean_verify_sonar:
                sonar-Dsonar.projectKey=
                weather-forecast-Dsonar.
                projectName='weather-forecast'
                "
        }

        timeout(time:1, unit:'HOURS'){
            def qg = waitForQualityGate()
            if (qg.status != 'OK') {
                error "Quality_Gate_did_not_
                    pass._Check_the_SonarQube_
                    dashboard_for_details."
            }
        }

        bat 'mvn_clean_install'
    }
}

stage('Build_reservation-service') {
    steps {

```



```

script{
    dir('JEEMicroServices/microService-
        reservation'){

        withSonarQubeEnv('sonarserver') {
            bat "mvn_clean_verify_sonar:
                sonar-Dsonar.projectKey=
                weather-forecast-Dsonar.
                projectName='weather-forecast'
                "
        }
        timeout(time:1, unit:'HOURS'){
            def qg = waitForQualityGate()
            if (qg.status != 'OK') {
                error "Quality_Gate_did_not_
                    pass._Check_the_SonarQube_
                    dashboard_for_details."
            }
        }

        bat 'mvn_clean_install'
    }
}

```

```

stage('Build_salle-service') {
    steps {
        script{
            dir('JEEMicroServices/microService-
                salle'){

                withSonarQubeEnv('sonarserver') {
                    bat "mvn_clean_verify_sonar:
                        sonar-Dsonar.projectKey=
                        weather-forecast-Dsonar.
                        projectName='weather-forecast'
                        "
                }
                timeout(time:1, unit:'HOURS'){
                    def qg = waitForQualityGate()
                    if (qg.status != 'OK') {
                        error "Quality_Gate_did_not_
                            pass._Check_the_SonarQube_
                            dashboard_for_details."
                    }
                }
            }

            bat 'mvn_clean_install'
        }
    }
}

```

```

}

stage('Build_emploiDuTemp-service') {
    steps {
        script{
            dir('JEEMicroServices/microService-
            emploiDuTemp'){

                withSonarQubeEnv('sonarserver') {
                    bat "mvn_clean_verify_sonar:
                        sonar-Dsonar.projectKey=
                        weather-forecast-Dsonar.
                        projectName='weather-forecast'
                        "
                }
            }
            timeout(time:1, unit:'HOURS'){
                def qg = waitForQualityGate()
                if (qg.status != 'OK') {
                    error "Quality_Gate_did_not_
                        pass._Check_the_SonarQube_
                        dashboard_for_details."
                }
            }
        }

        bat 'mvn_clean_install'
    }
}

```

```

    }

    }

}

stage('Build_Etudiant-service') {
    steps {
        script{
            dir('JEEMicroServices/microService-
                Etudiant'){

                withSonarQubeEnv('sonarserver') {
                    bat "mvn_clean_verify_sonar:
                        sonar-Dsonar.projectKey=
                        weather-forecast-Dsonar.
                        projectName='weather-forecast'
                        "
                }

                timeout(time:1, unit:'HOURS'){
                    def qg = waitForQualityGate()
                    if (qg.status != 'OK') {
                        error "Quality_Gate_did_not_
                            pass._Check_the_SonarQube_
                            dashboard_for_details."
                    }
                }
            }
        }
    }
}

```

```

        bat 'mvn_clean_install'
    }
}

}

stage('Build_Cours-service') {
    steps {
        script{
            dir('JEEMicroServices/microService-
                Cours'){

                withSonarQubeEnv('sonarserver') {
                    bat "mvn_clean_verify_sonar:
                        sonar-Dsonar.projectKey=
                        weather-forecast-Dsonar.
                        projectName='weather-forecast'
                        "
                }

                timeout(time:1, unit:'HOURS'){
                    def qg = waitForQualityGate()
                    if (qg.status != 'OK') {
                        error "Quality_Gate_did_not_
                            pass._Check_the_SonarQube_
                            dashboard_for_details."
                    }
                }
            }
        }
    }
}

```

```

    }

    bat 'mvn_clean_install'

    }

}

stage('Build_Absence-service') {
steps {
script{
    dir('JEEMicroServices/microService-
        Absence') {

        withSonarQubeEnv('sonarserver') {
            bat "mvn_clean_verify_sonar:
                sonar-Dsonar.projectKey=
                weather-forecast-Dsonar.
                projectName='weather-forecast'
                "
        }

        timeout(time:1, unit:'HOURS'){
            def qg = waitForQualityGate()
            if (qg.status != 'OK') {
                error "Quality_Gate_did_not_
                    pass._Check_the_SonarQube_"
            }
        }
    }
}
}

```

```

        dashboard_for_details."
    }
}

bat 'mvn_clean_install'
}
}

}

}

stage('Build_Supp_ui') {
    steps {
        dir('JEEMicroServicesFront') {
            bat 'npm_install'
        }
    }
}

stage('Compose_and_Build_Docker_Images') {
    steps {

        bat 'docker_compose_up_--build_d'

    }
}

```

```

    }

    }

}

```

Declarative: Tool Install	Git Clone	Build Discovery- Service	Build gateway- service	Build PROF- service	Build group- service	Build paiement- service	Build reservation- service	Build salle- service	Build emploiDuTemp- service	Build Etudiant- service	Build Cours- service	Build Absence- service	Build Supp ui	Compose and Build Docker Images
288ms	4s	139s	113s	94s	92s	78s	70s	73s	71s	64s	70s	73s	69s	105s
288ms	4s	139s	113s	94s	92s	78s	70s	73s	71s	64s	70s	73s	69s	105s

- Conclusion

En conclusion, les réalisations accomplies jusqu'à présent ont établi des fondations solides pour le projet. Les progrès réalisés dans la création d'une application de gestion des heures supplémentaires utilisant l'architecture de microservices ont démontré la viabilité et l'efficacité de cette méthode. Cependant, le travail continue. L'amélioration continue de l'application, l'ajout de nouvelles fonctionnalités et l'optimisation des performances sont des perspectives futures. En outre, l'accent sera mis sur

adapter l'application aux besoins changeants des utilisateurs et aux changements technologiques. En somme, bien qu'il y ait eu beaucoup de progrès, il reste encore beaucoup à faire pour atteindre pleinement la vision du projet. Nous espérons relever ces défis à l'avenir.