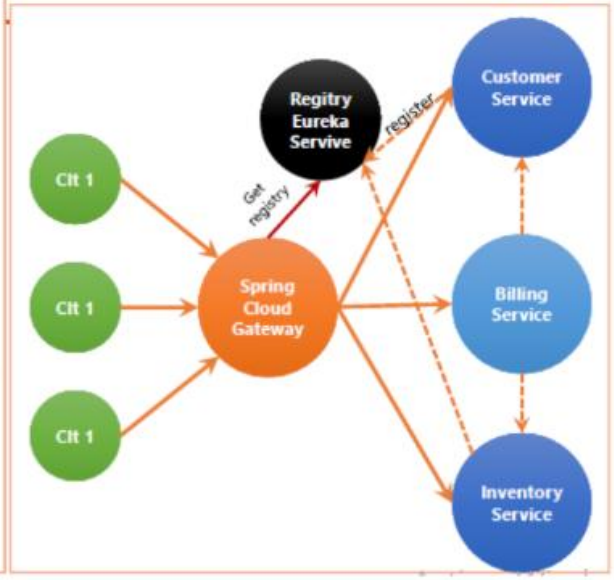
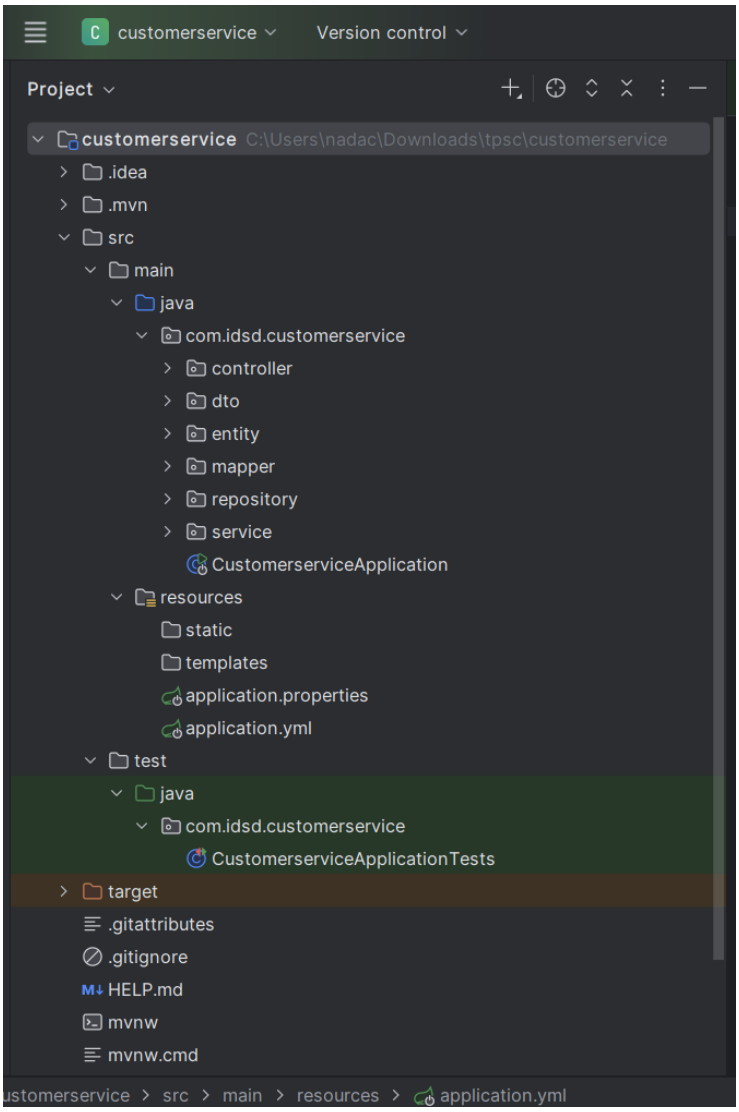
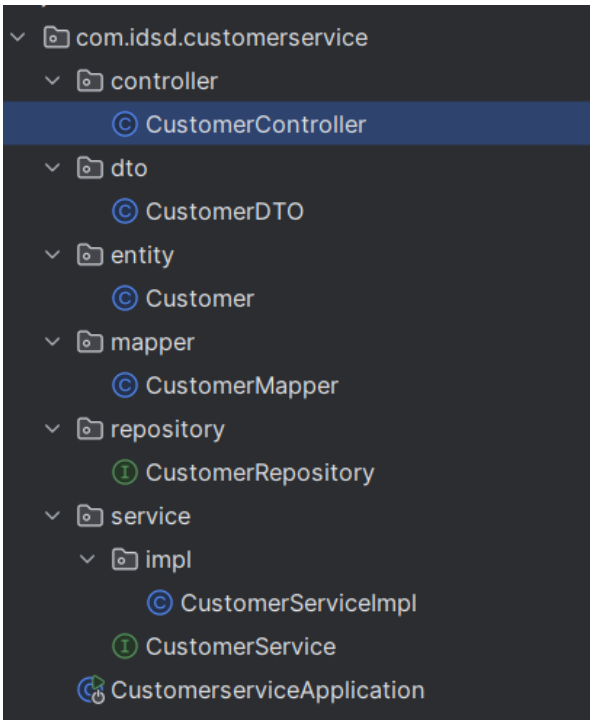
Activité pratique N° : Architectures Micro-services avec Spring cloud

Activité Pratique : Travail à faire

1. Créer le micro service Customer-service
   - Créer l'entité Customer
   - Créer l'interface CustomerRepository basée sur Spring Data
   - Déployer l'API Restful du micro-service en utilisant Spring Data Rest
   - Tester le Micro service
2. Créer le micro service Inventory-service
   - Créer l'entité Product
   - Créer l'interface ProductRepository basée sur Spring Data
   - Déployer l'API Restful du micro-service en utilisant Spring Data Rest
   - Tester le Micro service
3. Créer la Gateway service en utilisant Spring Cloud Gateway
   1. Tester la Service proxy en utilisant une configuration Statique basée sur le fichier application.yml
   2. Tester la Service proxy en utilisant une configuration Statique basée une configuration Java
4. Créer l'annuaire Registry Service basé sur NetFlix Eureka Server
5. Tester le proxy en utilisant une configuration dynamique de Gestion des routes vers les micro services enregistrés dans l'annuaire Eureka Server
6. Créer Le service Billing-Service en utilisant Open Feign pour communiquer avec les services Customer-service et Inventory-service
7. Créer un client Angular qui permet d'afficher une facture




customerservice
Version control
Project
customerservice C:\Users\nadac\Downloads\tpsc\customerservice
.idea
.mvn
src
main
java
com.idsd.customerservice
controller
dto
entity
mapper
repository
service
CustomerserviceApplication
resources
static
templates
application.properties
application.yml
test
java
com.idsd.customerservice
CustomerserviceApplicationTests
target
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
ustomerservice › src › main › resources › application.yml


com.idsd.customerservice
controller
CustomerController
dto
CustomerDTO
entity
Customer
mapper
CustomerMapper
repository
CustomerRepository
service
impl
CustomerServiceImpl
CustomerService
CustomerserviceApplication

```java
package com.idsd.customerservice.controller;
import com.idsd.customerservice.dto.CustomerDTO;
import com.idsd.customerservice.service.CustomerService;
import lombok.RequiredArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.List;
@RestController
@RequestMapping("/api/customers")
@RequiredArgsConstructor
public class CustomerController {
    private final CustomerService customerService;
    @PostMapping
    public ResponseEntity<CustomerDTO> create(@RequestBody CustomerDTO dto) {
        return ResponseEntity.status(HttpStatus.CREATED).body(customerService.createCustomer(dto));
    }
    @GetMapping
    public List<CustomerDTO> getAll() { return customerService.getAllCustomers(); }

    @GetMapping("/{id}")
    public ResponseEntity<CustomerDTO> getById(@PathVariable Long id) {
        return ResponseEntity.ok(customerService.getCustomerById(id));
    }
    @DeleteMapping("/{id}")
    public ResponseEntity<Void> delete(@PathVariable Long id) {
        customerService.deleteCustomer(id);
        return ResponseEntity.noContent().build();
    }
}
```

```java
package com.idsd.customerservice.dto;
import lombok.*;

@Data  36 usages
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class CustomerDTO {
    private Long id;
    private String name;
    private String email;
}
```

```java
package com.idsd.customerservice.entity;
import jakarta.persistence.*;
import lombok.*;
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Customer {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;
    private String email;
}
```

```java
package com.idsd.customerservice.mapper;

import com.idsd.customerservice.dto.CustomerDTO;
import com.idsd.customerservice.entity.Customer;

public class CustomerMapper {  5 usages
    public static CustomerDTO toDto(Customer customer) {  3 usages
        return CustomerDTO.builder()
                .id(customer.getId())
                .name(customer.getName())
                .email(customer.getEmail())
                .build();
    }

    public static Customer toEntity(CustomerDTO dto) {  1 usage
        return Customer.builder()
                .id(dto.getId())
                .name(dto.getName())
                .email(dto.getEmail())
                .build();
    }
}
```

```java
package com.idsd.customerservice.repository;

import com.idsd.customerservice.entity.Customer;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CustomerRepository extends JpaRepository<Customer, Long> {
}
```

```java
package com.idsd.customerservice.service.impl;
import com.idsd.customerservice.dto.CustomerDTO;
import com.idsd.customerservice.entity.Customer;
import com.idsd.customerservice.mapper.CustomerMapper;
import com.idsd.customerservice.repository.CustomerRepository;
import com.idsd.customerservice.service.CustomerService;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;
import java.util.List;
import java.util.stream.Collectors;
@Service
@RequiredArgsConstructor
public class CustomerServiceImpl implements CustomerService {
    private final CustomerRepository repository;
    @Override  1 usage
    public CustomerDTO createCustomer(CustomerDTO dto) {
        Customer customer = CustomerMapper.toEntity(dto);
        return CustomerMapper.toDto(repository.save(customer));
    }
    @Override  1 usage
    public List<CustomerDTO> getAllCustomers() {
        return repository.findAll().stream()  Stream<Customer>
                .map(CustomerMapper::toDto)  Stream<CustomerDTO>
                .collect(Collectors.toList());
    }
    @Override  1 usage
    public CustomerDTO getCustomerById(Long id) {
        return repository.findById(id)  Optional<Customer>
                .map(CustomerMapper::toDto)  Optional<CustomerDTO>
                .orElseThrow(() -> new RuntimeException("Customer not found"));
    }
    @Override  1 usage
```

```java
package com.idsd.customerservice.service;

import com.idsd.customerservice.dto.CustomerDTO;
import java.util.List;

public interface CustomerService {  4 usages  1 implementation
    CustomerDTO createCustomer(CustomerDTO dto);  1 usage  1 implementation
    List<CustomerDTO> getAllCustomers();  1 usage  1 implementation
    CustomerDTO getCustomerById(Long id);  1 usage  1 implementation
    void deleteCustomer(Long id);  1 usage  1 implementation
}
```

```java
package com.idsd.customerservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class CustomerserviceApplication {

    public static void main(String[] args) {
        SpringApplication.run(CustomerserviceApplication.class, args);
    }
}
```

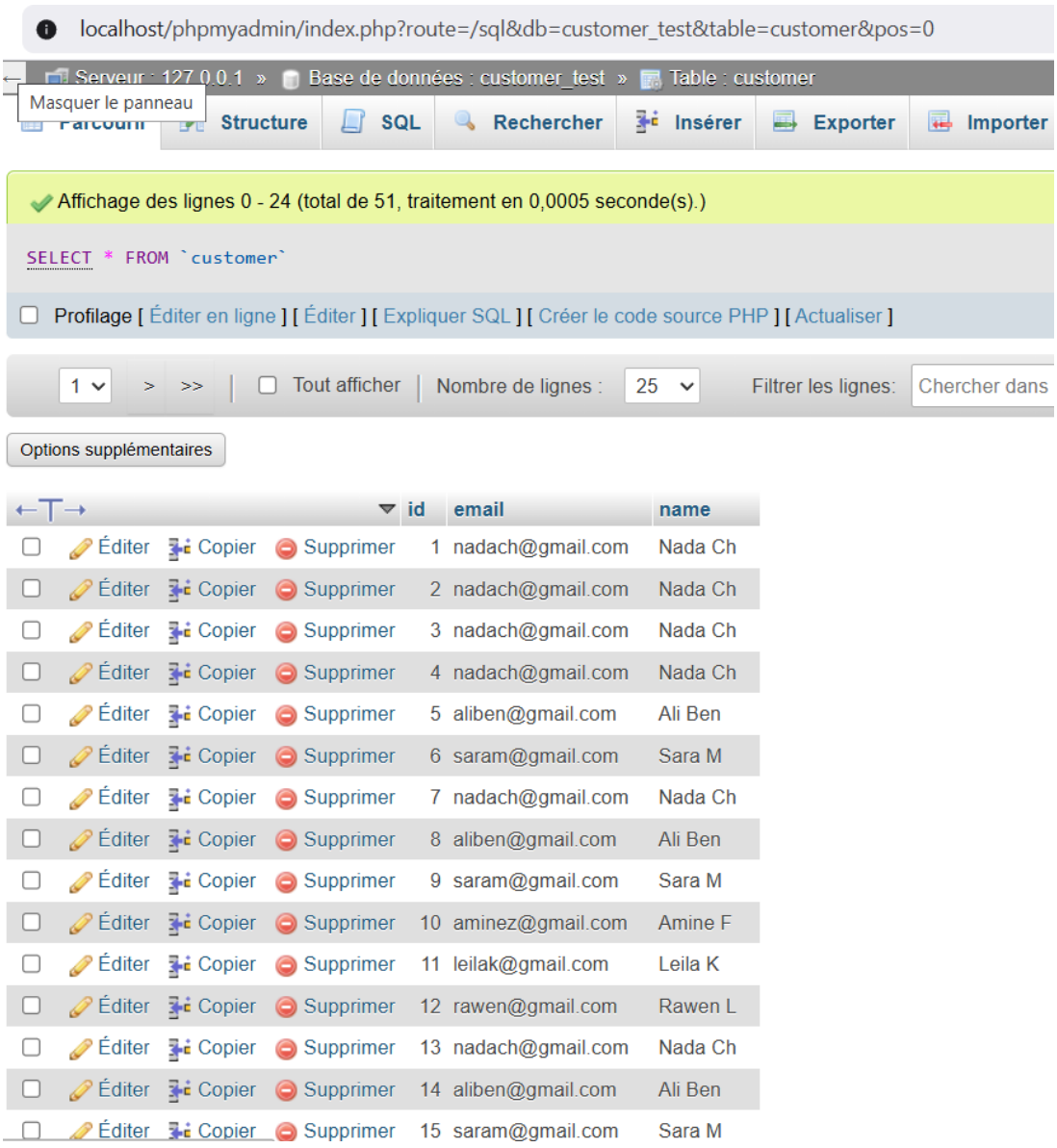
```java
package com.idsd.customerservice;
import com.idsd.customerservice.dto.CustomerDTO;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.boot.test.web.client.TestRestTemplate;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.test.context.ActiveProfiles;
import org.springframework.transaction.annotation.Transactional;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import static org.junit.jupiter.api.Assertions.*;
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
@ActiveProfiles("test")
public class CustomerserviceApplicationTests {
    @Autowired
    private TestRestTemplate restTemplate;
    private List<Long> customerIds = new ArrayList<>();  3 usages
    private Long customerId;  3 usages
    @BeforeEach
    @Transactional
    public void setUp() {
        String[][] customersData = {
                {"Nada Ch", "nadach@gmail.com"},
                {"Ali Ben", "aliben@gmail.com"},
                {"Sara M", "saram@gmail.com"}
        };
        for (String[] data : customersData) {
            CustomerDTO customerDTO = CustomerDTO.builder()
```

```java
    @Test
    public void testCreateCustomer() {
        CustomerDTO customerDTO = CustomerDTO.builder()
                .name("New Client")
                .email("newclient@gmail.com")
                .build();

        ResponseEntity<CustomerDTO> response = restTemplate.postForEntity( url: "/api/customers", customerDTO, CustomerDTO.class);

        assertEquals(HttpStatus.CREATED, response.getStatusCode());
        assertNotNull(response.getBody().getId());
    }

    @Test
    public void testGetCustomerById() {
        assertNotNull(customerId,  message: "L'ID du client créé doit être non nul");

        ResponseEntity<CustomerDTO> response = restTemplate.getForEntity( url: "/api/customers/" + customerId, CustomerDTO.class);

        assertEquals(HttpStatus.OK, response.getStatusCode());
        assertNotNull(response.getBody(),  message: "Le client récupéré ne doit pas être nul");
    }

    @Test
    public void testCreateMultipleCustomers() {
        String[][] customersData = {
                {"Amine F", "aminez@gmail.com"},
                {"Leila K", "leilak@gmail.com"},
                {"Rawen L", "rawen@gmail.com"}
```

```java
@Test
public void testCreateMultipleCustomers() {
    String[][] customersData = {
            {"Amine F", "aminez@gmail.com"},
            {"Leila K", "leilak@gmail.com"},
            {"Rawen L", "rawen@gmail.com"}
    };

    for (String[] data : customersData) {
        CustomerDTO customerDTO = CustomerDTO.builder()
                .name(data[0])
                .email(data[1])
                .build();

        ResponseEntity<CustomerDTO> response = restTemplate.postForEntity( url: "/api/customers", customerDTO, CustomerDTO.class);

        assertEquals(HttpStatus.CREATED, response.getStatusCode());
        assertNotNull(response.getBody().getId(), message: "L'ID du client créé doit être non nul");
    }
}

@Test
public void testGetAllCustomers() {
    ResponseEntity<CustomerDTO[]> response = restTemplate.getForEntity( url: "/api/customers", CustomerDTO[].class);

    assertEquals(HttpStatus.OK, response.getStatusCode());
    assertNotNull(response.getBody());
    assertTrue( condition: response.getBody().length >= customerIds.size(), message: "La liste des clients ne doit pas être vide");
}
}
```

```properties
# Application Name
spring.application.name=customerservice

# MySQL Database Configuration
spring.datasource.url=jdbc:mysql://localhost:3306/customer_test?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=

# JPA/Hibernate Configuration
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

# Thymeleaf Configuration
spring.thymeleaf.cache=false

# Eureka Client Configuration
eureka.client.service-url.defaultZone=http://localhost:8761/eureka
eureka.client.fetch-registry=true
eureka.client.register-with-eureka=true
```

Enregistrement dans la base :



	id	email	name
Éditer Copier Supprimer	1	nadach@gmail.com	Nada Ch
Éditer Copier Supprimer	2	nadach@gmail.com	Nada Ch
Éditer Copier Supprimer	3	nadach@gmail.com	Nada Ch
Éditer Copier Supprimer	4	nadach@gmail.com	Nada Ch
Éditer Copier Supprimer	5	aliben@gmail.com	Ali Ben
Éditer Copier Supprimer	6	saram@gmail.com	Sara M
Éditer Copier Supprimer	7	nadach@gmail.com	Nada Ch
Éditer Copier Supprimer	8	aliben@gmail.com	Ali Ben
Éditer Copier Supprimer	9	saram@gmail.com	Sara M
Éditer Copier Supprimer	10	aminez@gmail.com	Amine F
Éditer Copier Supprimer	11	leilak@gmail.com	Leila K
Éditer Copier Supprimer	12	rawen@gmail.com	Rawen L
Éditer Copier Supprimer	13	nadach@gmail.com	Nada Ch
Éditer Copier Supprimer	14	aliben@gmail.com	Ali Ben
Éditer Copier Supprimer	15	saram@gmail.com	Sara M