Réalisé par : Amina Dahmouni

Classe : II-BDCC3

Encadré par : Pr Mohamed YOUSSFI

[Compte Rendu]

Mise en œuvre d'une architecture micro-services

Systèmes Distribués et Big Data Processing

Customer-Service

Couche DAO (Entités JPA et Repositories)

Entite Customer :

```java
import javax.persistence.Entity;
import javax.persistence.Id;


@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class Customer {
    @Id
    private String id;
    private String name;
    private String email;
}
```

CustomerRepository :

```java
package com.example.customerservice.repositories;

import com.example.customerservice.entities.Customer;
import org.springframework.data.jpa.repository.JpaRepository;

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

La couche DTO

CustomerRequestDTO :

```java
package com.example.customerservice.dto;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data @NoArgsConstructor @AllArgsConstructor
public class CustomerRequestDTO {
    private String id;
    private String name;
    private String email;
}
```

CustomerRespenseDTO :

```java
package com.example.customerservice.dto;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data @NoArgsConstructor @AllArgsConstructor
public class CustomerResponseDTO {
    private String id;
    private String name;
    private String email;
}
```

Mappers (DTO <=>Entities)

CustomerMapper :

```java
package com.example.customerservice.mappers;

import com.example.customerservice.dto.CustomerRequestDTO;
import com.example.customerservice.dto.CustomerResponseDTO;
import com.example.customerservice.entities.Customer;
import org.mapstruct.Mapper;

@Mapper(componentModel = "spring")
public interface CustomerMapper {
    CustomerResponseDTO customerToCustomerResponseDTO(Customer customer);
    Customer CustomerRequestDTOTocustomer(CustomerRequestDTO customerRequestDTO);
}
```

Couche Service

CustomerService :

```java
package com.example.customerservice.services;

import com.example.customerservice.dto.CustomerRequestDTO;
import com.example.customerservice.dto.CustomerResponseDTO;
import java.util.List;

public interface CustomerService {
    CustomerResponseDTO save(CustomerRequestDTO customerRequestDTO);
    CustomerResponseDTO getCustomer(String id);
    CustomerResponseDTO update(CustomerRequestDTO customerRequestDTO);
    List<CustomerResponseDTO> listCustomers();

}
```

La couche Web (Rest Controllers)

```java
        @RestController
        @RequestMapping(path = "/api")
        public class CustomerRestAPI {
            private CustomerService customerService;

            public CustomerRestAPI(CustomerService customerService) {
                this.customerService = customerService;
            }
            @GetMapping(path = "/customers")
            public List<CustomerResponseDTO> allCustomers(){
                return customerService.listCustomers();
            }
            @PostMapping(path = "/customers")
            public CustomerResponseDTO save(@RequestBody CustomerRequestDTO customerRequestDTO){
                customerRequestDTO.setId(UUID.randomUUID().toString());
                return customerService.save(customerRequestDTO);
            }
            @GetMapping(path = "/customers/{id}")
            public CustomerResponseDTO getCustomer(@PathVariable String id){
                return customerService.getCustomer(id);
            }
        }
```

Insertion des clients dans la base de données

```java
package com.example.customerservice;

import ...

@SpringBootApplication
public class CustomerServiceApplication {

    public static void main(String[] args) { SpringApplication.run(CustomerServiceApplication.class, args); }
    @Bean
    CommandLineRunner start(CustomerService customerService){
        return args -> {
            customerService.save(new CustomerRequestDTO( id: "A01", name: "Amina", email: "amina@gmail.com"));
            customerService.save(new CustomerRequestDTO( id: "A02", name: "Fatima", email: "fatima@gmail.com"));
        };
    }
}
```

Configuration de l'application


```
1    server.port=8082
2    spring.application.name=CUSTOMER-SERVICE
3    spring.h2.console.enabled=true
4    #pas besoin de s'enregister dans discovery service
5    spring.cloud.discovery.enabled=false
6    spring.datasource.url=jdbc:h2:mem:customer-db
7
```

Création de la base de données :

SELECT * FROM CUSTOMER;

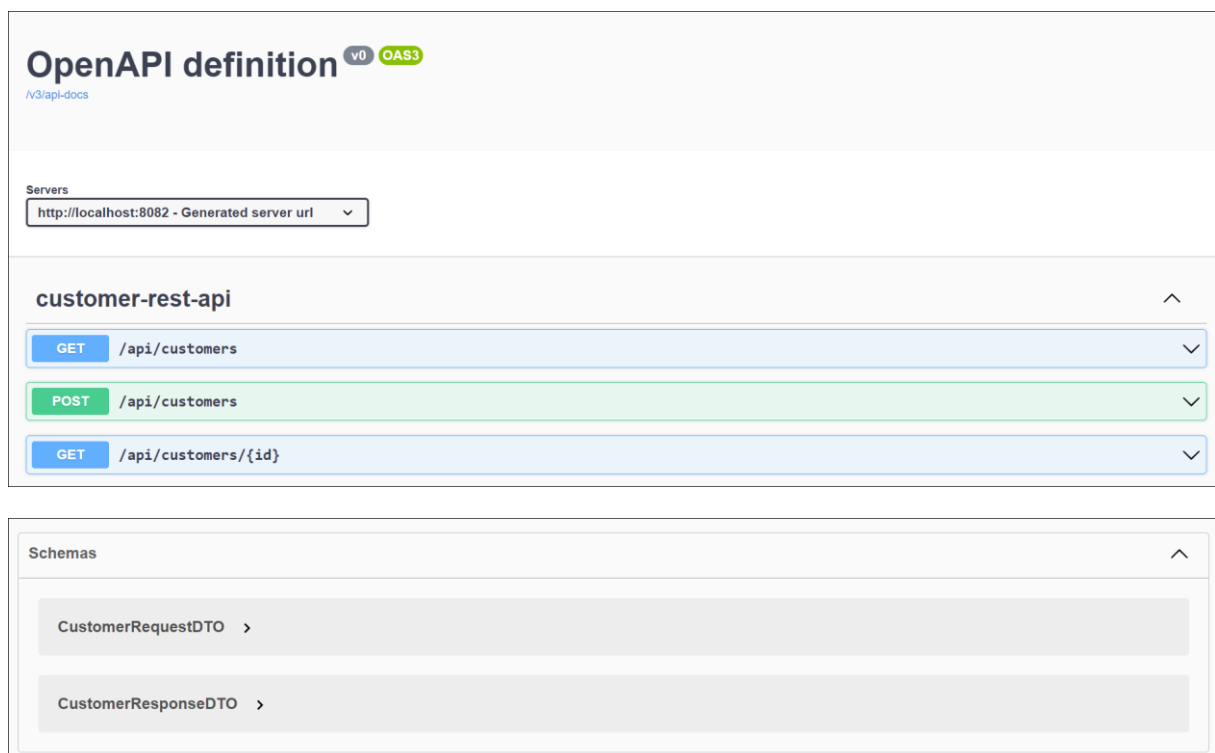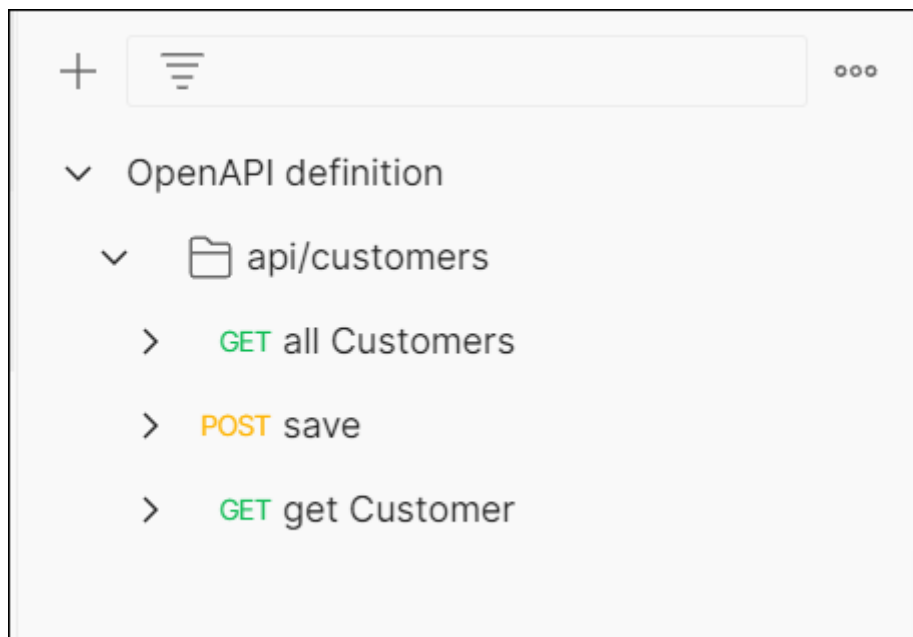| ID	EMAIL	NAME
A01	amina@gmail.com	Amina
A02	fatima@gmail.com	Fatima

(2 rows, 14 ms)

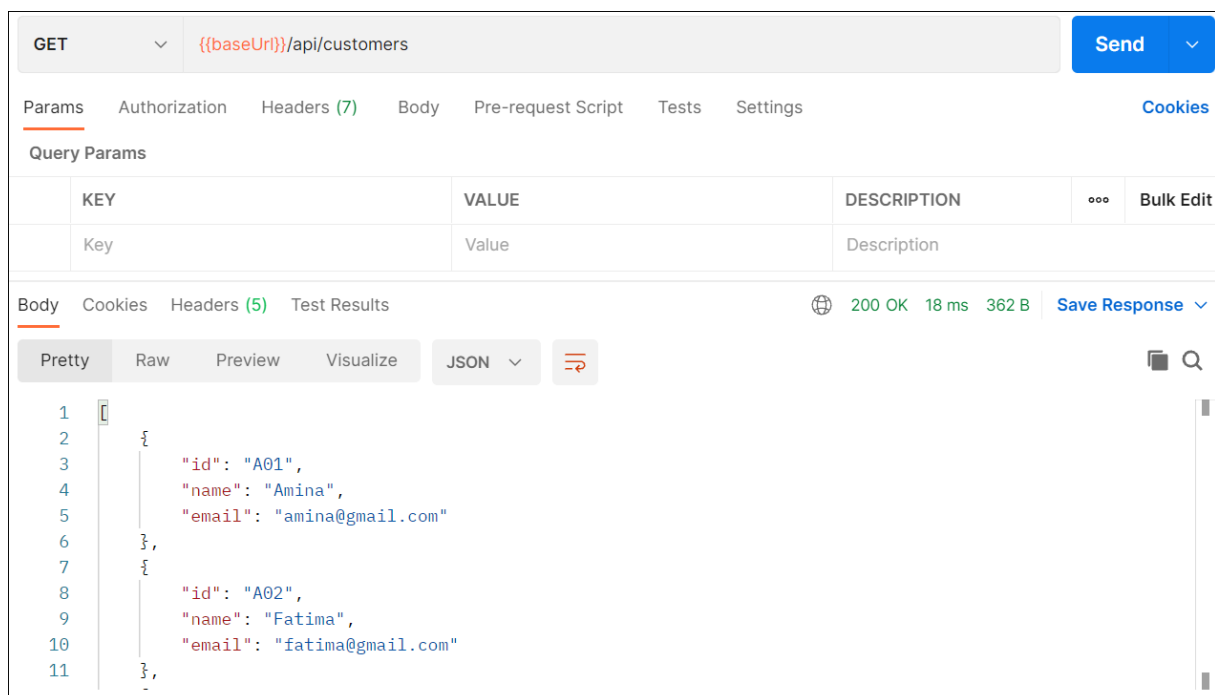## La structure du projet :
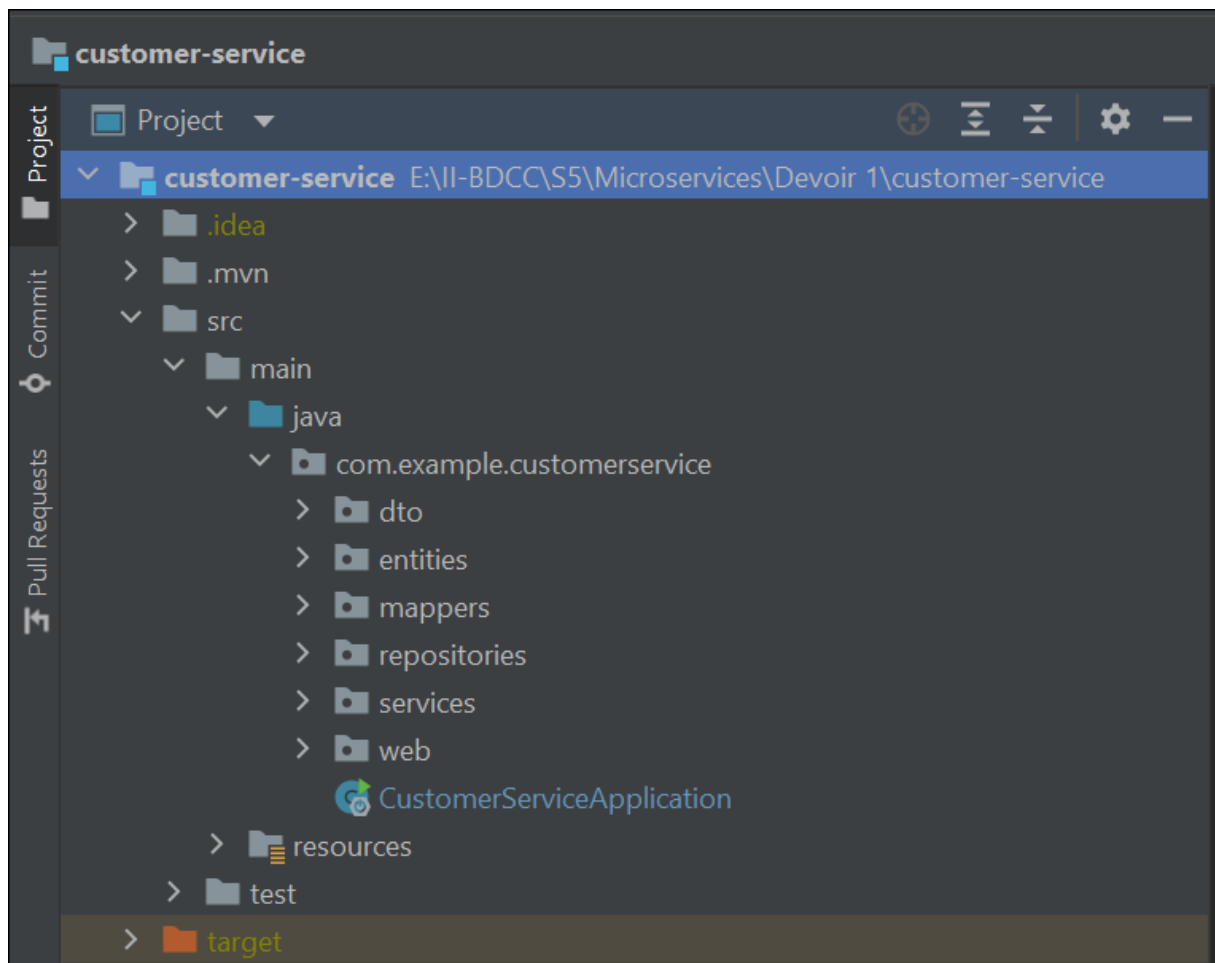
La documentation du service avec Open API

Importer la documentation dans Postman :

Afficher la liste des clients :

Chercher un client par ID :

GET `{{baseUrl}}/api/customers/A01` **Send**

Params | Authorization | Headers (7) | Body | Pre-request Script | Tests | Settings **Cookies**

Query Params

KEY	VALUE	DESCRIPTION		Bulk Edit

Body | Cookies | Headers (5) | Test Results   200 OK 15 ms 217 B **Save Response**

Pretty | Raw | Preview | Visualize   JSON

```
1  {
2      "id": "A01",
3      "name": "Amina",
4      "email": "amina@gmail.com"
5  }
```

Ajouter un client :

POST `{{baseUrl}}/api/customers` **Send**

Params | Authorization | Headers (10) | Body ● | Pre-request Script | Tests | Settings **Cookies**

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL JSON **Beautify**

```
1  {
2      "name": "Safae",
3      "email": "safae@gmail.com"
4  }
```

Body | Cookies | Headers (5) | Test Results   200 OK 10 ms 250 B **Save Response**

Pretty | Raw | Preview | Visualize   JSON

```
1  {
2      "id": "3773e602-067f-4255-8ed8-fcb8dbd486c6",
3      "name": "Safae",
4      "email": "safae@gmail.com"
5  }
```

```
1  [
2      {
3          "id": "A01",
4          "name": "Amina",
5          "email": "amina@gmail.com"
6      },
7      {
8          "id": "A02",
9          "name": "Fatima",
10         "email": "fatima@gmail.com"
11     },
12     {
13         "id": "8504e0ed-abdb-449e-a6e0-38a4b8d3d0be",
14         "name": "Safae",
15         "email": "safae@gmail.com"
16     },
17     {
```

Billing-Service

Couche DAO (Entités JPA et Repositories)

Entitie Invoice :

```java
package com.example.billingservice.entities;

import ...
@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class Invoice {
    @Id
    private String id;
    private Date date;
    private BigDecimal amount;
    private String customerID;
    @Transient
    private Customer customer;
}
```

```java
package com.example.billingservice.models;

import ...

@Data @NoArgsConstructor @AllArgsConstructor
public class Customer {
    private String id;
    private String name;
    private String email;
}
```

InvoiceRepository :

```java
package com.example.billingservice.repositories;

import com.example.billingservice.entities.Invoice;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface InvoiceRepository extends JpaRepository<Invoice,String> {

    List<Invoice> findByCustomerID(String clientID);
}
```

La couche DTO

InvoiceRequest :

```java
package com.example.billingservice.dto;
import ...

@Data @NoArgsConstructor @AllArgsConstructor
public class InvoiceRequestDTO {
    private BigDecimal amount;
    private String customerID;
}
```

InvoiceResponse :

```java
package com.example.billingservice.dto;

import ...

@Data @NoArgsConstructor @AllArgsConstructor
public class InvoiceResponseDTO {
    private String id;
    private Date date;
    private BigDecimal amount;
    private Customer customer;
}
```

Mappers (DTO <=>Entities)

InvoiceMapper :

```java
package com.example.billingservice.mappers;

import ...

@Mapper(componentModel = "spring")
public interface InvoiceMapper {
    Invoice fromInvoiceRequestDTO(InvoiceRequestDTO invoiceRequestDTO);
    InvoiceResponseDTO fromInvoice(Invoice invoice);
}
```

Couche Service

InvoiceService :

```java
package com.example.billingservice.services;

import ...

public interface InvoiceService {
    InvoiceResponseDTO save(InvoiceRequestDTO invoiceRequestDTO);
    InvoiceResponseDTO getInvoice(String invoiceId);
    List<InvoiceResponseDTO> invoicesByCustomerId(String customerId);
    List<InvoiceResponseDTO> allInvoices();
}
```

La couche Web (Rest Controllers)

InvoiceRestController :

```java
@RestController
@RequestMapping(path = "/api")
public class InvoiceRestController {
    private InvoiceService invoiceService;

    public InvoiceRestController(InvoiceService invoiceService) { this.invoiceService = invoiceService;
    @GetMapping(path = "/invoices/{id}")
    public InvoiceResponseDTO getInvoice(@PathVariable(name = "id") String invoiceId){
        return invoiceService.getInvoice(invoiceId);
    }
    @GetMapping(path = "/invoicesByCustomer/{customerID}")
    public List<InvoiceResponseDTO> getInvoicesByCustomer(@PathVariable String customerID){
        return invoiceService.invoicesByCustomerId(customerID);
    }
    @PostMapping(path = "/invoices")
    public InvoiceResponseDTO save(@RequestBody InvoiceRequestDTO invoiceRequestDTO){
        return invoiceService.save(invoiceRequestDTO);
    }
    @GetMapping(path = "/invoices")
    public List<InvoiceResponseDTO> allInvoices() { return invoiceService.allInvoices(); }
    @ExceptionHandler(Exception.class)
    public ResponseEntity<String> exceptionHandler(Exception e){
        return new ResponseEntity<>(e.getMessage(), HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

Insertion des clients dans la base de données

```java
@SpringBootApplication
@EnableFeignClients
public class BillingServiceApplication {

    public static void main(String[] args) { SpringApplication.run(BillingServiceApplication.class, args); }
    @Bean
    CommandLineRunner commandLineRunner(InvoiceService invoiceService){
        return args -> {
            invoiceService.save(new InvoiceRequestDTO(BigDecimal.valueOf(76000), customerID: "A01"));
            invoiceService.save(new InvoiceRequestDTO(BigDecimal.valueOf(34000), customerID: "A01"));
            invoiceService.save(new InvoiceRequestDTO(BigDecimal.valueOf(54000), customerID: "A02"));
        };
    }
}
```
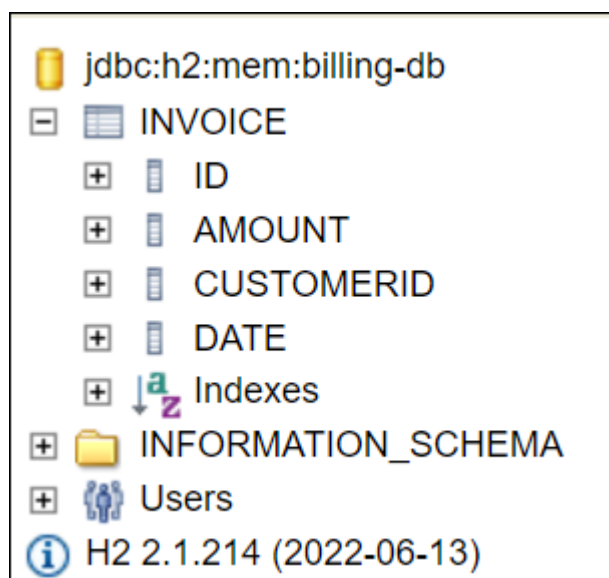
Configuration de l'application
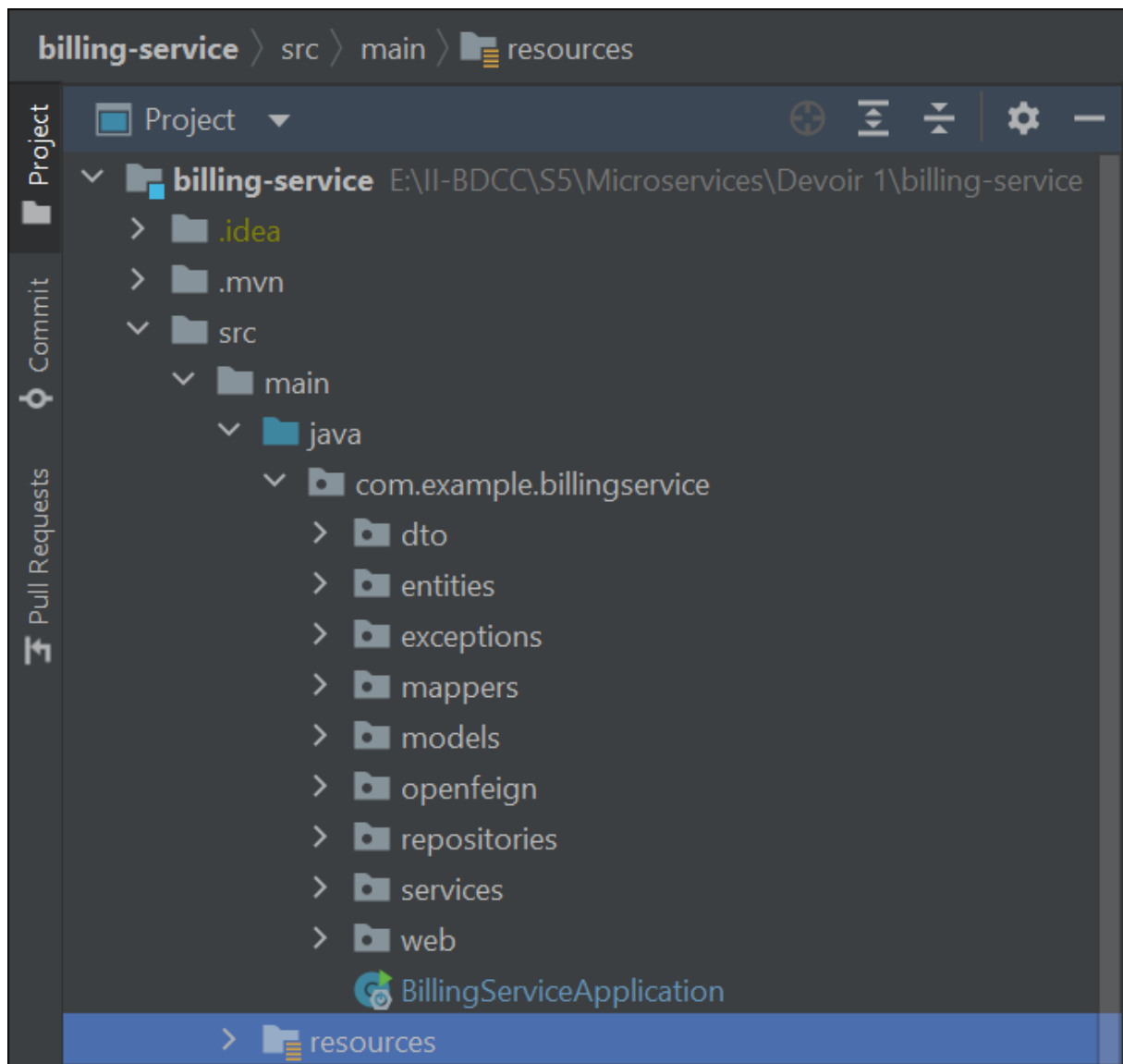
```properties
application.properties ×

Plugins supporting application.properties files found.

1   server.port=8083
2   spring.application.name=BILLING-SERVICE
3   spring.h2.console.enabled=true
4   #pas besoin de s'enregister dans discovery service
5   spring.cloud.discovery.enabled=true
6   spring.datasource.url=jdbc:h2:mem:billing-db
7   eureka.instance.prefer-ip-address=true
8
```

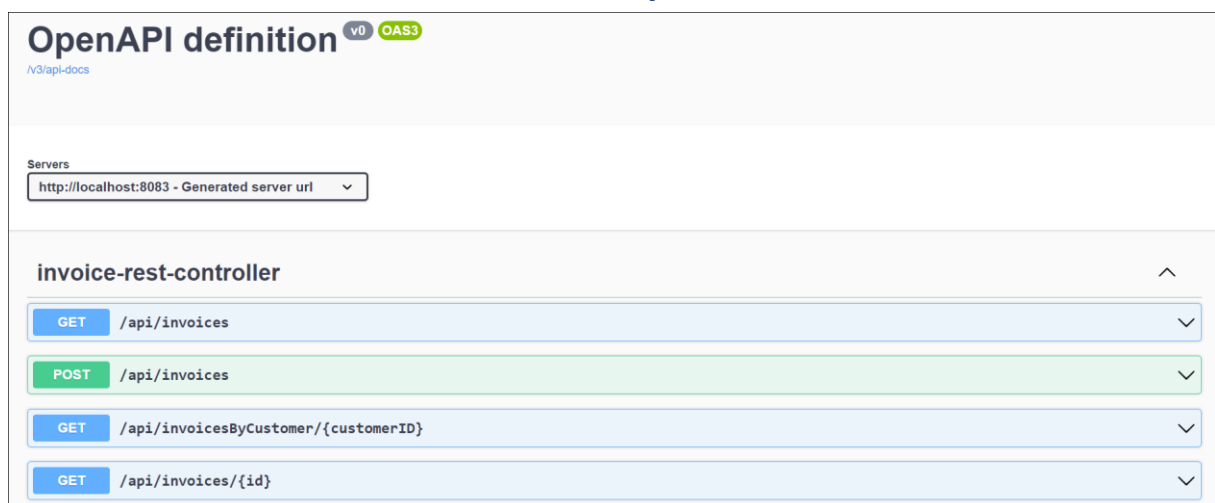## Création de la base de données :

```
jdbc:h2:mem:billing-db
  INVOICE
    ID
    AMOUNT
    CUSTOMERID
    DATE
    Indexes
  INFORMATION_SCHEMA
  Users
  H2 2.1.214 (2022-06-13)
```
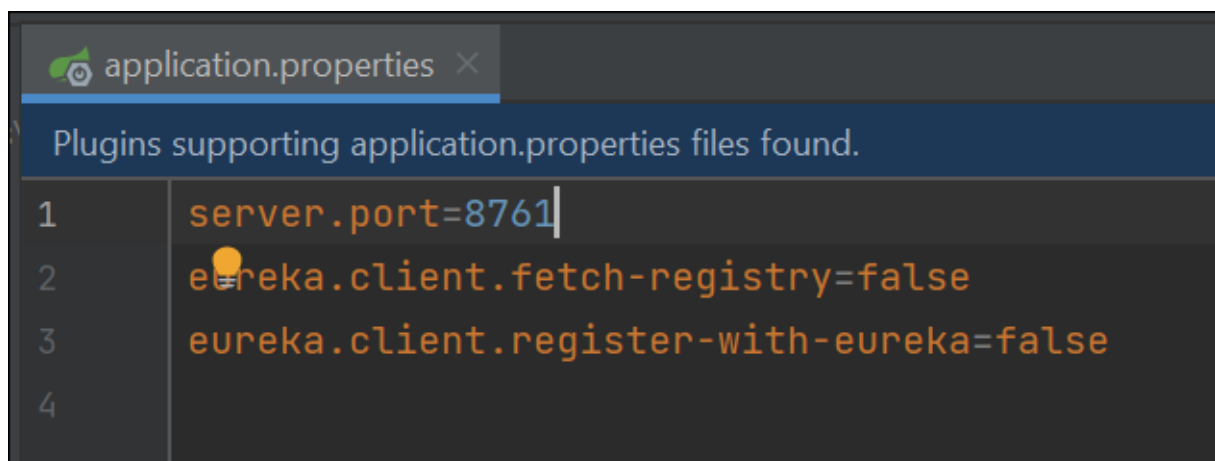
La structure du projet :

La documentation du service avec Open API

Schemas   ^

InvoiceRequestDTO >

Customer >

InvoiceResponseDTO >

Eureka-Service


```java
package com.example.eurekaservice;

import ...

@SpringBootApplication
@EnableEurekaServer
public class EurekaServiceApplication {

    public static void main(String[] args) {

        SpringApplication.run(EurekaServiceApplication.class, args);
    }

}
```


application.properties ×

Plugins supporting application.properties files found.

```properties
server.port=8761
eureka.client.fetch-registry=false
eureka.client.register-with-eureka=false
```

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
BILLING-SERVICE	n/a (1)	(1)	UP (1) - LAPTOP-B5P9C1HH:BILLING-SERVICE:8083
CUSTOMER-SERVICE	n/a (1)	(1)	UP (1) - LAPTOP-B5P9C1HH:CUSTOMER-SERVICE:8082

Gateway-Service

GatewayServiceApplication.java   application.properties   pom.xml (gateway-service)

```java
package com.example.gatewayservice;

import ...

@SpringBootApplication
public class GatewayServiceApplication {

    public static void main(String[] args) { SpringApplication.run(GatewayServiceApplication.class, args); }
    @Bean
    DiscoveryClientRouteDefinitionLocator discoveryClientRouteDefinitionLocator(
            ReactiveDiscoveryClient rdc, DiscoveryLocatorProperties dlp
            ){
        return new DiscoveryClientRouteDefinitionLocator(rdc,dlp);
    }
}
```

GatewayServiceApplication.java   application.properties   pom.xml (gateway-service)

Plugins supporting application.properties files found.

```properties
server.port=9999
spring.application.name=GATEWAY
spring.cloud.discovery.enabled=true
eureka.instance.prefer-ip-address=true
```
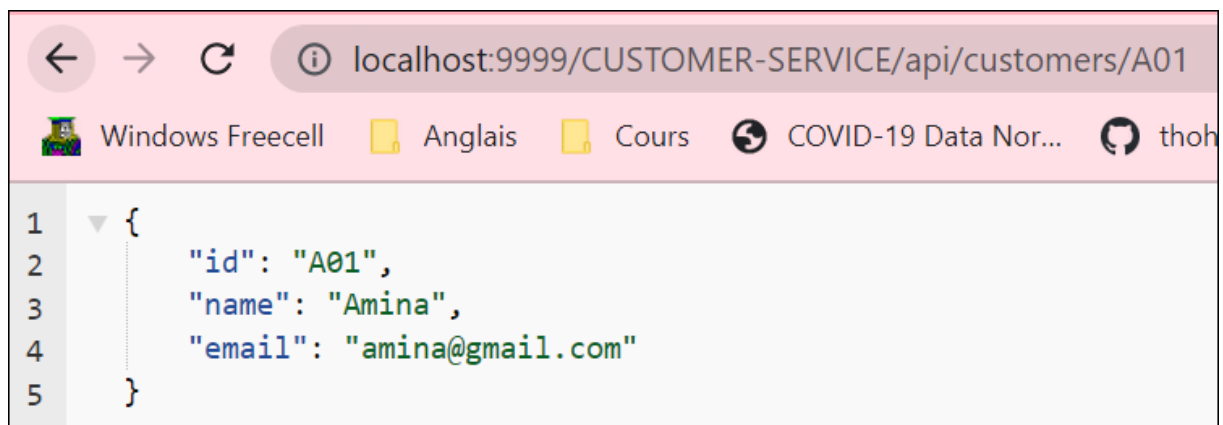
Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
BILLING-SERVICE	n/a (1)	(1)	UP (1) - LAPTOP-B5P9C1HH:BILLING-SERVICE:8083
CUSTOMER-SERVICE	n/a (1)	(1)	UP (1) - LAPTOP-B5P9C1HH:CUSTOMER-SERVICE:8082
GATEWAY	n/a (1)	(1)	UP (1) - LAPTOP-B5P9C1HH:GATEWAY:9999

Afficher la liste des Clients :

```
localhost:9999/CUSTOMER-SERVICE/api/customers/

1   [
3       {
4           "id": "A01",
5           "name": "Amina",
6           "email": "amina@gmail.com"
7       },
9       {
10          "id": "A02",
11          "name": "Fatima",
12          "email": "fatima@gmail.com"
13      }
14  ]
```
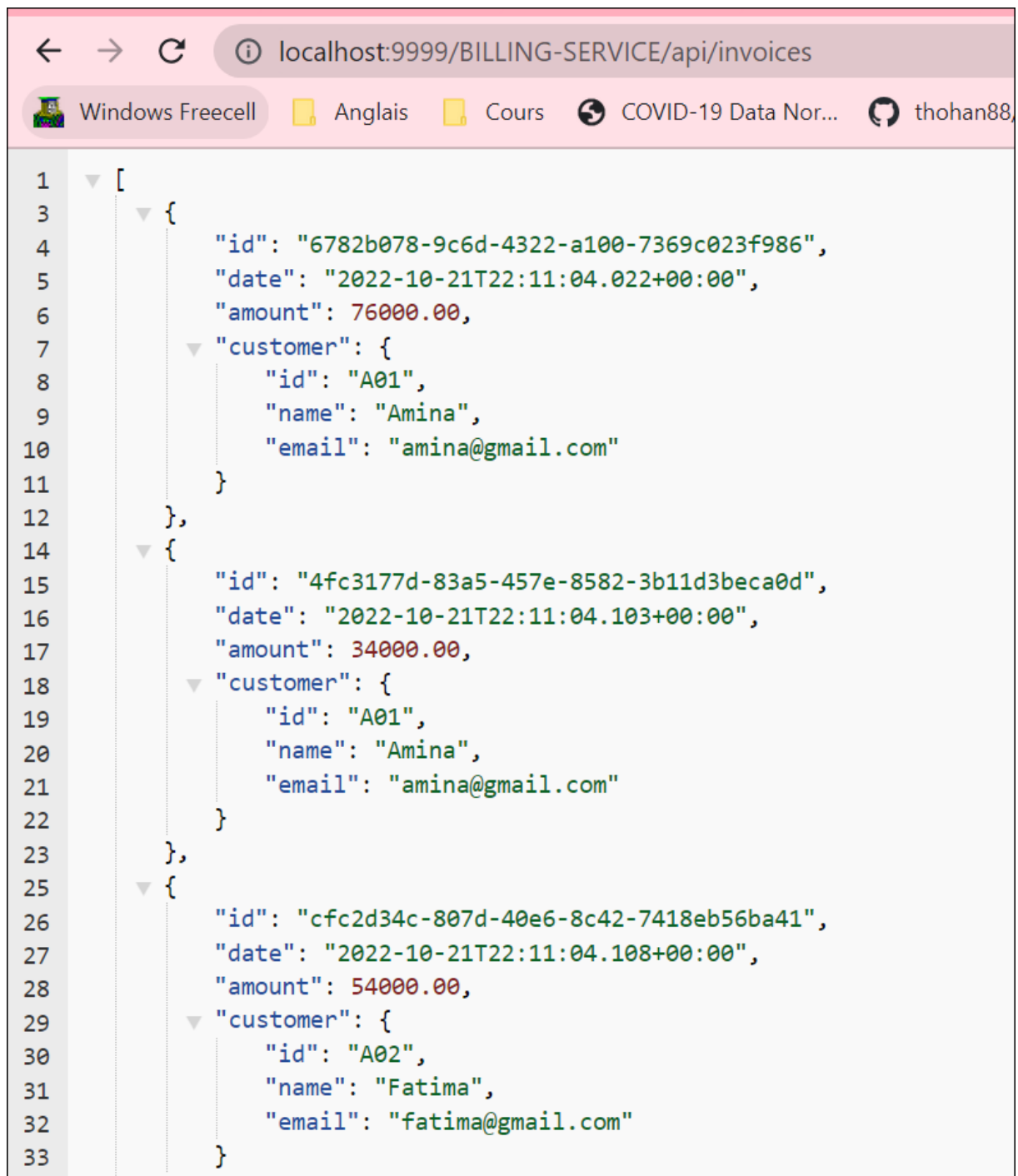
Chercher un client par ID :

```
localhost:9999/CUSTOMER-SERVICE/api/customers/A01

1   {
2       "id": "A01",
3       "name": "Amina",
4       "email": "amina@gmail.com"
5   }
```

Afficher la liste des factures :

```
localhost:9999/BILLING-SERVICE/api/invoices

Windows Freecell      Anglais      Cours      COVID-19 Data Nor...      thohan88/

1   [
3       {
4           "id": "6782b078-9c6d-4322-a100-7369c023f986",
5           "date": "2022-10-21T22:11:04.022+00:00",
6           "amount": 76000.00,
7           "customer": {
8               "id": "A01",
9               "name": "Amina",
10              "email": "amina@gmail.com"
11          }
12      },
14      {
15          "id": "4fc3177d-83a5-457e-8582-3b11d3beca0d",
16          "date": "2022-10-21T22:11:04.103+00:00",
17          "amount": 34000.00,
18          "customer": {
19              "id": "A01",
20              "name": "Amina",
21              "email": "amina@gmail.com"
22          }
23      },
25      {
26          "id": "cfc2d34c-807d-40e6-8c42-7418eb56ba41",
27          "date": "2022-10-21T22:11:04.108+00:00",
28          "amount": 54000.00,
29          "customer": {
30              "id": "A02",
31              "name": "Fatima",
32              "email": "fatima@gmail.com"
33          }
```

POST http://localhost:9999/BILLING-SERVICE/api/invoices?

Params ●   Authorization   Headers (9)   **Body** ●   Pre-request Script   Tests   Settings

○ none   ○ form-data   ○ x-www-form-urlencoded   ● raw   ○ binary   ○ GraphQL   **JSON** ∨

```
1  {
2      "amount":65070,
3      "customerID":"A01"
4  }
```

Body   Cookies   Headers (3)   Test Results   ⊕ 200 OK   112 ms   280 B

Pretty   Raw   Preview   Visualize   JSON ∨

```
1  {
2      "id": "ac9c4aa3-d183-4d3f-96ac-4f9f0e2d2915",
3      "date": "2022-10-21T22:12:05.672+00:00",
4      "amount": 65070,
5      "customer": {
6          "id": "A01",
7          "name": "Amina",
8          "email": "amina@gmail.com"
9      }
```

GET http://localhost:9999/BILLING-SERVICE/api/invoices/ac9c4aa3-d183-4d3f-96ac-4f9f0e2d2915

Params   Authorization   Headers (9)   **Body** ●   Pre-request Script   Tests   Settings

○ none   ○ form-data   ○ x-www-form-urlencoded   ● raw   ○ binary   ○ GraphQL   **JSON** ∨

```
1  {
2      "amount":65000,
3      "customerID":"A01"
4  }
```

Body   Cookies   Headers (3)   Test Results   ⊕ 200 OK   135 ms   283 B

Pretty   Raw   Preview   Visualize   JSON ∨

```
1   {
2       "id": "ac9c4aa3-d183-4d3f-96ac-4f9f0e2d2915",
3       "date": "2022-10-21T22:12:05.672+00:00",
4       "amount": 65070.00,
5       "customer": {
6           "id": "A01",
7           "name": "Amina",
8           "email": "amina@gmail.com"
9       }
10  }
```

Ajouter une facture à un client qui n'existe pas :

POST   http://localhost:9999/BILLING-SERVICE/api/invoices?

Params ●   Authorization   Headers (9)   Body ●   Pre-request Script   Tests   Settings

○ none   ○ form-data   ○ x-www-form-urlencoded   ● raw   ○ binary   ○ GraphQL   **JSON** ∨

```
1  {
2     "amount":65070,
3     "customerID":"Blabal"
4  }
```

Body   Cookies   Headers (3)   Test Results   500 Internal Server Error   136 ms   153 B

Pretty   Raw   Preview   Visualize   Text ∨

```
1  Customer not found
```

Il génère l'exception CustomerNotFound.