

Réalisé par : Amina Dahmouni

Classe : II-BDCC3

Encadré par : Pr Mohamed YOUSSEFI



[Compte Rendu]

Mise en œuvre d'une architecture micro-services

Systemes Distribués et Big Data Processing



Customer-Service

Couche DAO (Entités JPA et Repositories)

Entite Customer :

```
7      import javax.persistence.Entity;
8      import javax.persistence.Id;
9
10     @Entity
11     @Data @NoArgsConstructor @AllArgsConstructor
12     public class Customer {
13         @Id
14         private String id;
15         private String name;
16         private String email;
17     }
18
```

CustomerRepository :

```
1      package com.example.customerservice.repositories;
2
3      import com.example.customerservice.entities.Customer;
4      import org.springframework.data.jpa.repository.JpaRepository;
5
6      public interface CustomerRepository extends JpaRepository<Customer,String> {
7      }
```

La couche DTO

CustomerRequestDTO :

```
1      package com.example.customerservice.dto;
2
3      import lombok.AllArgsConstructor;
4      import lombok.Data;
5      import lombok.NoArgsConstructor;
6
7      @Data @NoArgsConstructor @AllArgsConstructor
8      public class CustomerRequestDTO {
9          private String id;
10         private String name;
11         private String email;
12     }
```

CustomerResponseDTO :

```
1      package com.example.customerservice.dto;
2
3      import lombok.AllArgsConstructor;
4      import lombok.Data;
5      import lombok.NoArgsConstructor;
6
7      @Data @NoArgsConstructor @AllArgsConstructor
8      public class CustomerResponseDTO {
9          private String id;
10         private String name;
11         private String email;
12     }
```

Mappers (DTO <=>Entities)

CustomerMapper :

```
1 package com.example.customerservice.mappers;
2
3 import com.example.customerservice.dto.CustomerRequestDTO;
4 import com.example.customerservice.dto.CustomerResponseDTO;
5 import com.example.customerservice.entities.Customer;
6 import org.mapstruct.Mapper;
7
8 @Mapper(componentModel = "spring")
9 public interface CustomerMapper {
10     CustomerResponseDTO customerToCustomerResponseDTO(Customer customer);
11     Customer CustomerRequestDTOToCustomer(CustomerRequestDTO customerRequestDTO);
12 }
```

Couche Service

CustomerService :

```
1 package com.example.customerservice.services;
2
3 import com.example.customerservice.dto.CustomerRequestDTO;
4 import com.example.customerservice.dto.CustomerResponseDTO;
5 import java.util.List;
6
7 public interface CustomerService {
8     CustomerResponseDTO save(CustomerRequestDTO customerRequestDTO);
9     CustomerResponseDTO getCustomer(String id);
10    CustomerResponseDTO update(CustomerRequestDTO customerRequestDTO);
11    List<CustomerResponseDTO> listCustomers();
12
13 }
```

La couche Web (Rest Controllers)

```
11  @RestController
12  @RequestMapping(path = "/api")
13  public class CustomerRestAPI {
14      private CustomerService customerService;
15
16      public CustomerRestAPI(CustomerService customerService) {
17          this.customerService = customerService;
18      }
19
20      @GetMapping(path = "/customers")
21      public List<CustomerResponseDTO> allCustomers(){
22          return customerService.listCustomers();
23      }
24
25      @PostMapping(path = "/customers")
26      public CustomerResponseDTO save(@RequestBody CustomerRequestDTO customerRequestDTO){
27          customerRequestDTO.setId(UUID.randomUUID().toString());
28          return customerService.save(customerRequestDTO);
29      }
30
31      @GetMapping(path = "/customers/{id}")
32      public CustomerResponseDTO getCustomer(@PathVariable String id){
33          return customerService.getCustomer(id);
34      }
35  }
```

Insertion des clients dans la base de données

```
CustomerServiceApplication.java  application.properties
1  package com.example.customerservice;
2
3  import ...
4
5
6
7
8
9
10 @SpringBootApplication
11 public class CustomerServiceApplication {
12
13     public static void main(String[] args) { SpringApplication.run(CustomerServiceApplication.class, args); }
14
15     @Bean
16     CommandLineRunner start(CustomerService customerService){
17         return args -> {
18             customerService.save(new CustomerRequestDTO( id: "A01", name: "Amina", email: "amina@gmail.com"));
19             customerService.save(new CustomerRequestDTO( id: "A02", name: "Fatima", email: "fatima@gmail.com"));
20         };
21     }
22 }
23 }
```

Configuration de l'application

```
application.properties x
Plugins supporting application.properties files found.
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 :

Auto commit ☒ Max rows: 1000

jdbc:h2:mem:customer-db

CUSTOMER

- ID
- EMAIL
- NAME
- Indexes

INFORMATION_SCHEMA

Users

H2 2.1.214 (2022-06-13)

Run Run Selected Auto complete

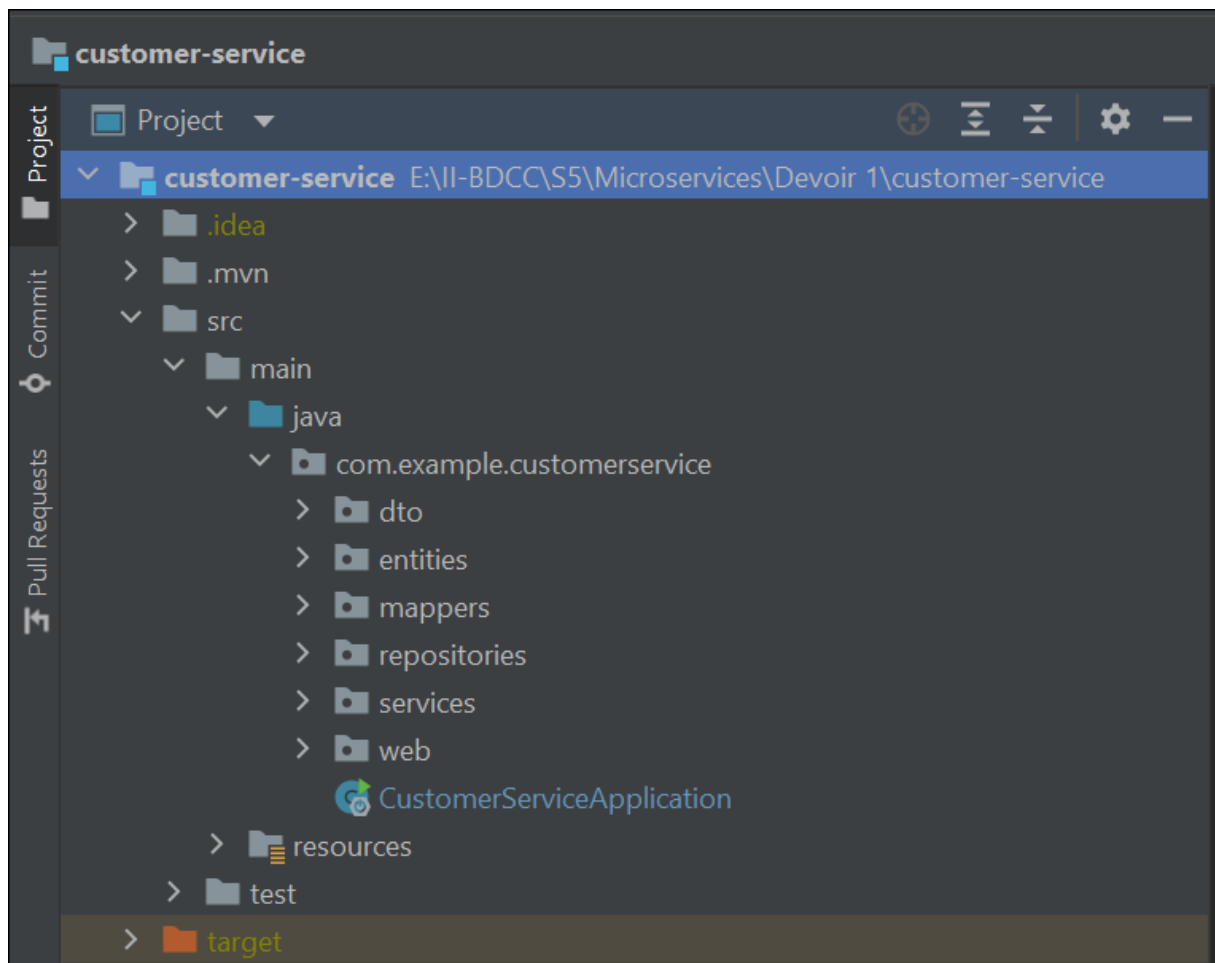
SELECT * FROM CUSTOMER |

SELECT * FROM CUSTOMER;

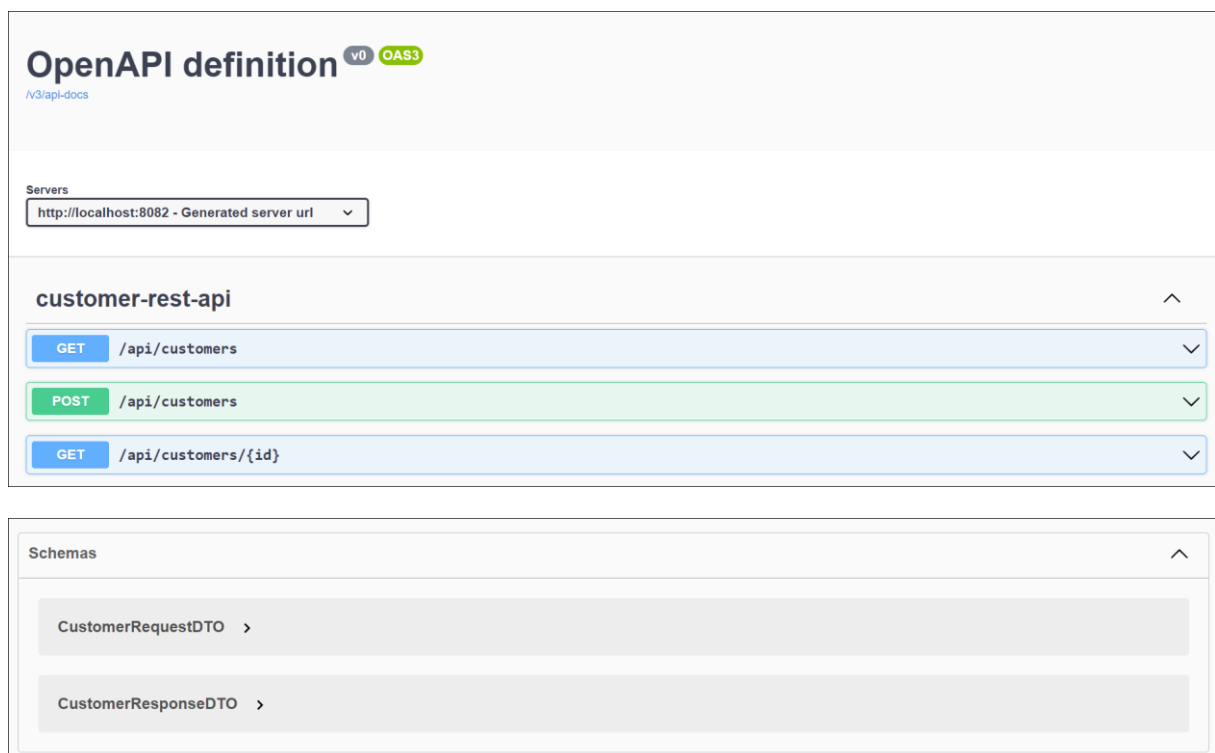
ID	EMAIL	NAME
A01	amina@gmail.com	Amina
A02	fatima@gmail.com	Fatima

(2 rows, 14 ms)

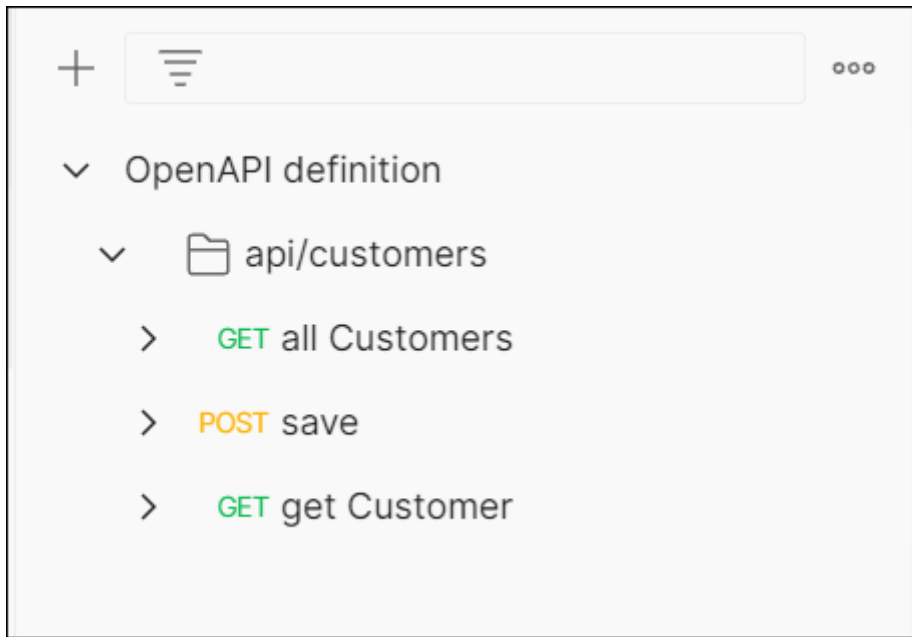
La structure du projet :



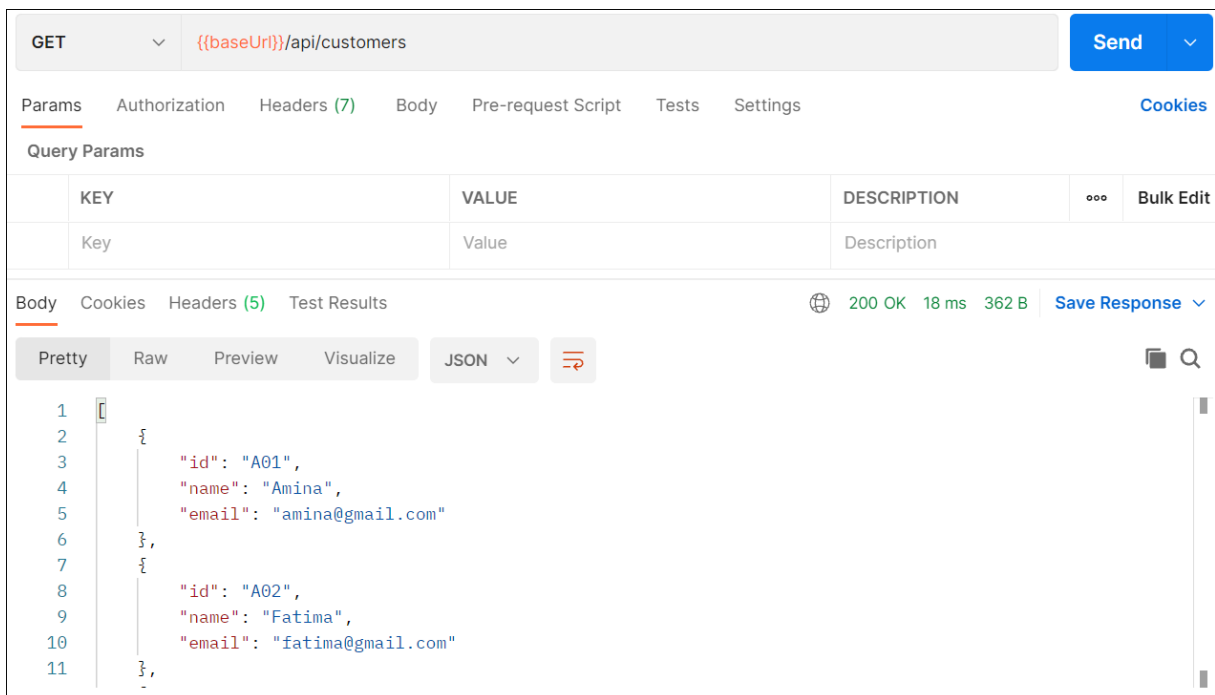
La documentation du service avec Open API



Importer la documentation dans Postman :



Afficher la liste des clients :



Chercher un client par ID :

The screenshot shows a REST client interface with a GET request to `{{baseUrl}}/api/customers/A01`. The response is a JSON object with the following structure:

```
{
  "id": "A01",
  "name": "Amina",
  "email": "amina@gmail.com"
}
```

The status bar indicates a 200 OK response with a 15 ms latency and 217 B of data.

Ajouter un client :

The screenshot shows a REST client interface with a POST request to `{{baseUrl}}/api/customers`. The request body is a JSON object with the following structure:

```
{
  "name": "Safae",
  "email": "safae@gmail.com"
}
```

The response is a JSON object with the following structure:

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

The status bar indicates a 200 OK response with a 10 ms latency and 250 B of data.

```
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)

Entité Invoice :

```
Invoice.java x
1 package com.example.billing.service.entities;
2
3 import ...
4
13 @Entity
14 @Data @NoArgsConstructor @AllArgsConstructor
15 public class Invoice {
16     @Id
17     private String id;
18     private Date date;
19     private BigDecimal amount;
20     private String customerID;
21     @Transient
22     private Customer customer;
23 }
```

```
Customer.java x
1 package com.example.billing.service.models;
2
3 import ...
4
6
7 @Data @NoArgsConstructor @AllArgsConstructor
8 public class Customer {
9     private String id;
10    private String name;
11    private String email;
12 }
```

InvoiceRepository :

```
InvoiceRepository.java x
1 package com.example.billing.service.repositories;
2
3 import com.example.billing.service.entities.Invoice;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 import java.util.List;
7
8 public interface InvoiceRepository extends JpaRepository<Invoice, String> {
9
10     List<Invoice> findByCustomerID(String clientID);
11 }
```

La couche DTO

InvoiceRequest :

```
InvoiceRequestDTO.java x
1 package com.example.billing.service.dto;
2 import ...
3
4 @Data @NoArgsConstructor @AllArgsConstructor
5 public class InvoiceRequestDTO {
6     private BigDecimal amount;
7     private String customerID;
8 }
9
10 }
```

InvoiceResponse :

```
InvoiceResponseDTO.java x
1  package com.example.billing.service.dto;
2
3  import ...
11
12  @Data @NoArgsConstructor @AllArgsConstructor
13  public class InvoiceResponseDTO {
14      private String id;
15      private Date date;
16      private BigDecimal amount;
17      private Customer customer;
18  }
```

Mappers (DTO <=>Entities)

InvoiceMapper :

```
InvoiceMapper.java x
1  package com.example.billing.service.mappers;
2
3  import ...
7
8  @Mapper(componentModel = "spring")
9  public interface InvoiceMapper {
10      Invoice fromInvoiceRequestDTO(InvoiceRequestDTO invoiceRequestDTO);
11      InvoiceResponseDTO fromInvoice(Invoice invoice);
12  }
```

Couche Service

InvoiceService :

```
InvoiceService.java x
1 package com.example.billing.service.services;
2
3 import ...
7
8 public interface InvoiceService {
9     InvoiceResponseDTO save(InvoiceRequestDTO invoiceRequestDTO);
10    InvoiceResponseDTO getInvoice(String invoiceId);
11    List<InvoiceResponseDTO> invoicesByCustomerId(String customerId);
12    List<InvoiceResponseDTO> allInvoices();
13 }
```

La couche Web (Rest Controllers)

InvoiceRestController :

```
InvoiceRestController.java x
13 @RestController
14 @RequestMapping(path = "/api")
15 public class InvoiceRestController {
16     private InvoiceService invoiceService;
17
18     public InvoiceRestController(InvoiceService invoiceService) { this.invoiceService = invoiceService; }
21     @GetMapping(path = "/invoices/{id}")
22     public InvoiceResponseDTO getInvoice(@PathVariable(name = "id") String invoiceId){
23         return invoiceService.getInvoice(invoiceId);
24     }
25     @GetMapping(path = "/invoicesByCustomer/{customerId}")
26     public List<InvoiceResponseDTO> getInvoicesByCustomer(@PathVariable String customerId){
27         return invoiceService.invoicesByCustomerId(customerId);
28     }
29     @PostMapping(path = "/invoices")
30     public InvoiceResponseDTO save(@RequestBody InvoiceRequestDTO invoiceRequestDTO){
31         return invoiceService.save(invoiceRequestDTO);
32     }
33     @GetMapping(path = "/invoices")
34     public List<InvoiceResponseDTO> allInvoices() { return invoiceService.allInvoices(); }
37     @ExceptionHandler(Exception.class)
38     @ResponseBody
39     public ResponseEntity<String> exceptionHandler(Exception e){
40         return new ResponseEntity<>(e.getMessage(), HttpStatus.INTERNAL_SERVER_ERROR);
41     }
42 }
```

Insertion des clients dans la base de données

```
@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"));
        };
    }
}
```

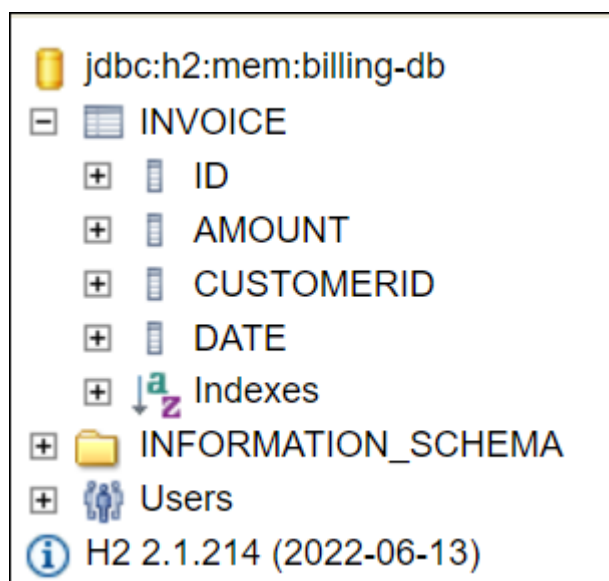
Configuration de l'application

application.properties

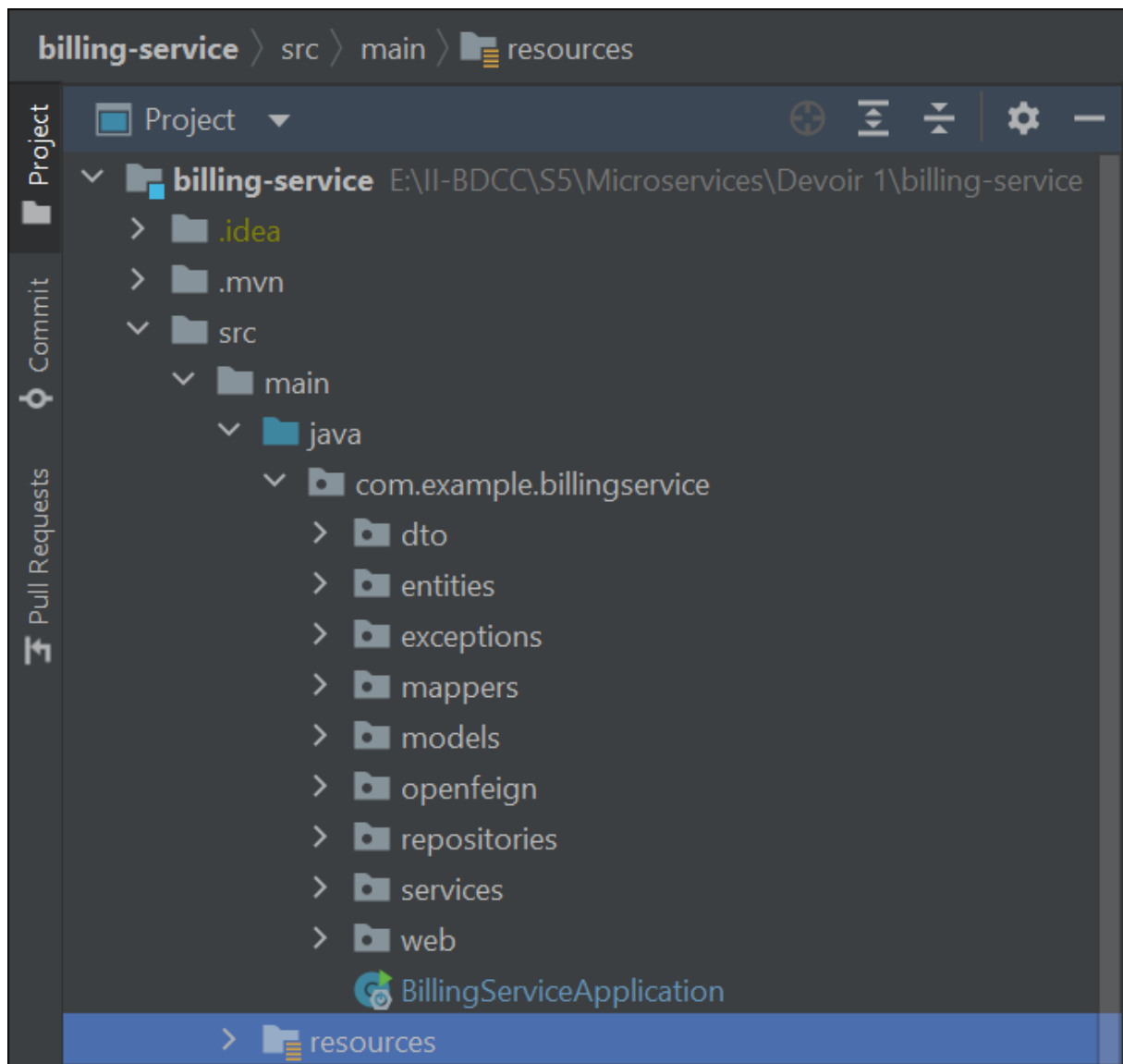
Plugins supporting application.properties files found.

```
1 server.port=8083
2 spring.application.name=BILLING-SERVICE
3 spring.h2.console.enabled=true
4 #pas besoin de s'enregistrer 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
```

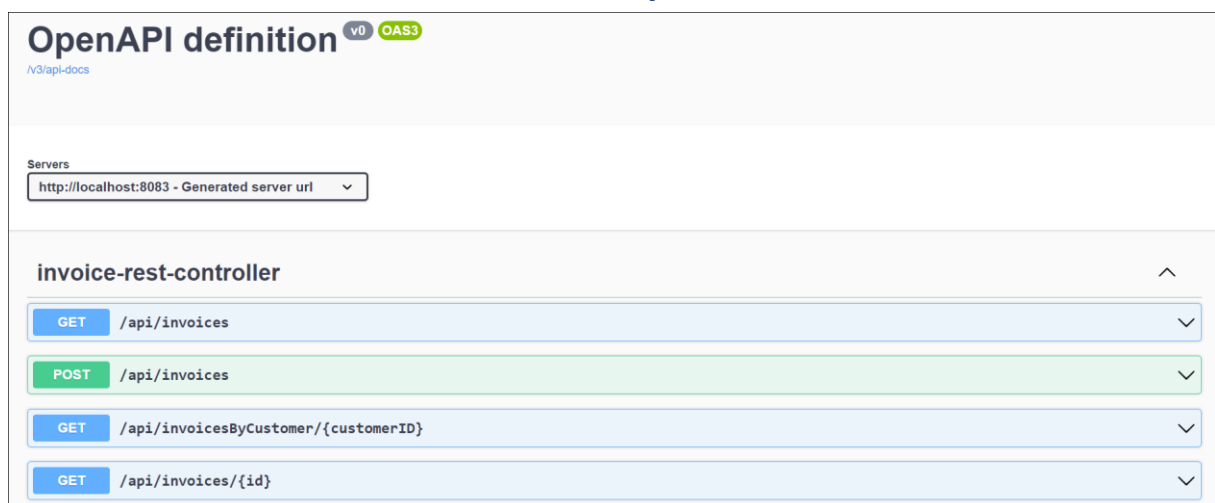
Création de la base de données :



La structure du projet :



La documentation du service avec Open API





Eureka-Service

```
EurekaServiceApplication.java x
1 package com.example.eurekaservice;
2
3 import ...
4
5
6
7 @SpringBootApplication
8 @EnableEurekaServer
9 public class EurekaServiceApplication {
10
11     public static void main(String[] args) {
12
13         SpringApplication.run(EurekaServiceApplication.class, args);
14     }
15
16 }
```

```
application.properties x
Plugins supporting application.properties files found.
1 server.port=8761
2 eureka.client.fetch-registry=false
3 eureka.client.register-with-eureka=false
4
```


localhost:8761

Windows Freecell Anglais Cours COVID-19 Data Nor... thohan88/covid19-... WordPress Themes... Bouffe – Restaurant... Menu Simple – Bou...

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)
1  package com.example.gatewayservice;
2
3  import ...
9
10 @SpringBootApplication
11 public class GatewayServiceApplication {
12
13     public static void main(String[] args) { SpringApplication.run(GatewayServiceApplication.class, args); }
16
17     @Bean
18     DiscoveryClientRouteDefinitionLocator discoveryClientRouteDefinitionLocator(
19         ReactiveDiscoveryClient rdc, DiscoveryLocatorProperties dlp
20     ){
21         return new DiscoveryClientRouteDefinitionLocator(rdc, dlp);
22     }
23 }

```

Plugins supporting application.properties files found.

```

1  server.port=9999
2  spring.application.name=GATEWAY
3  spring.cloud.discovery.enabled=true
4  eureka.instance.prefer-ip-address=true

```

Instances currently registered with Eureka

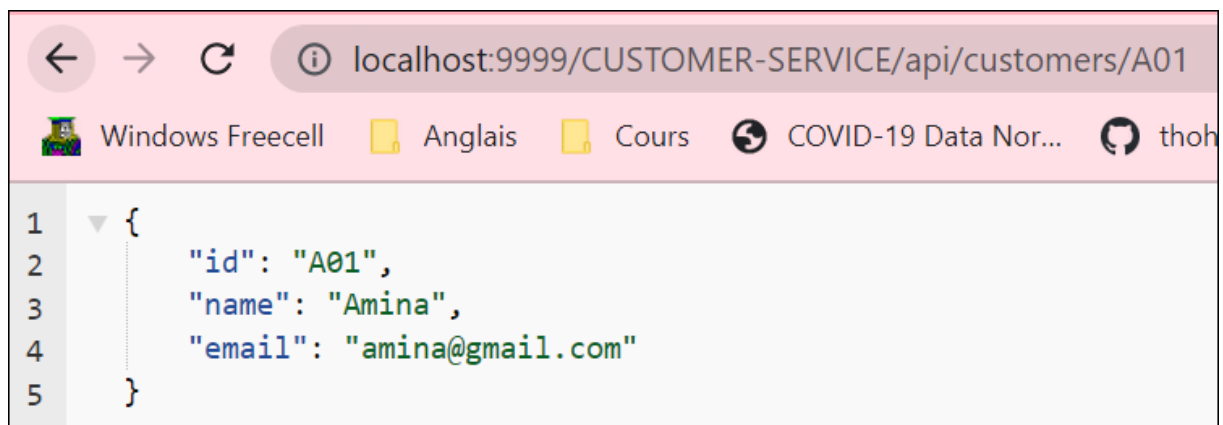
Application	AMIs	Availability Zones	Status
BILLING-SERVICE	n/a (1)	(1)	UP (1) - LAPTOP-B5P9C1HH:BILLING-SERVICE:8083
CUSTOMER-SERVICE	n/a (1)	(1)	UP (1) - LAPTOP-B5P9C1HH:CUSTOMER-SERVICE:8082
GATEWAY	n/a (1)	(1)	UP (1) - LAPTOP-B5P9C1HH:GATEWAY:9999

Afficher la liste des Clients :



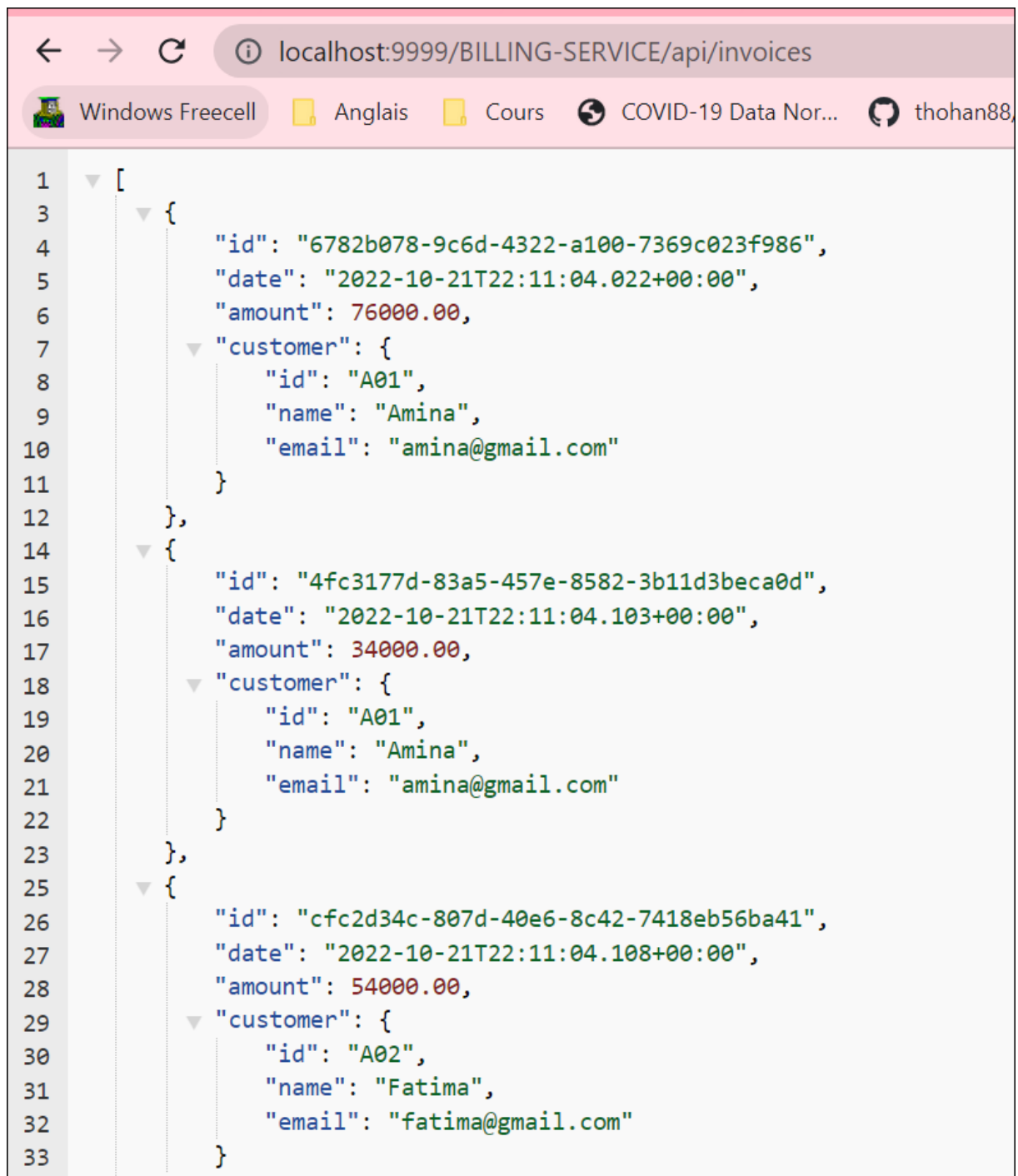
```
1  [
3    {
4      "id": "A01",
5      "name": "Amina",
6      "email": "amina@gmail.com"
7    },
9    {
10     "id": "A02",
11     "name": "Fatima",
12     "email": "fatima@gmail.com"
13   }
14 ]
```

Chercher un client par ID :



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

Afficher la liste des factures :



The screenshot shows a web browser window with the address bar displaying `localhost:9999/BILLING-SERVICE/api/invoices`. The browser's taskbar at the top includes icons for Windows Freecell, Anglais, Cours, COVID-19 Data Nor..., and a user profile named thohan88. The main content area displays a JSON response from a REST client, with line numbers 1 through 33 on the left. The JSON is a list of three invoice objects, each containing an id, date, amount, and a customer object with id, name, and email.

```
1  [
2
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    },
13
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    },
24
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      }
34    }
35  ]
```

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   }
}
```

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

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:9999/BILLING-SERVICE/api/invoices?
- Body:** JSON format with the following content:

```
1 {  
2   "amount": 65070,  
3   "customerID": "Blabal"  
4 }
```
- Response:** 500 Internal Server Error, 136 ms, 153 B
- Response Body:** Pretty view showing "1 Customer not found"

Il génère l'exception `CustomerNotFound`.

Déployer les micro services de démarrage à ressort dans un conteneur docker

Déployer customer-service :

```
PS E:\II-BDCC\S5\Microservices\Devoir 1\customer-service> docker build -t customer-service-web .  
Sending build context to Docker daemon 66.36MB  
Step 1/5 : FROM openjdk:17-alpine  
17-alpine: Pulling from library/openjdk  
5843afab3874: Pull complete  
53c9466125e4: Pull complete  
d8d715783b80: Pull complete  
Digest: sha256:4b6abae565492dbe9e7a894137c966a7485154238902f2f25e9dbd9784383d81  
Status: Downloaded newer image for openjdk:17-alpine
```

Déployer billing-service :

```

PS E:\II-BDCC\S5\Microservices\Devoir 1\billing-service> docker build -t billing-service-web .
Sending build context to Docker daemon 66.91MB
Step 1/5 : FROM openjdk:17-alpine
---> 264c9bdce361
Step 2/5 : VOLUME /tmp
---> Using cache
---> 50bedd188218
Step 3/5 : COPY target/billing-service-0.0.1-SNAPSHOT.jar /billing-service.jar
---> 1b9ffdc7570f
Step 4/5 : CMD ["java","-jar","/billing-service.jar","--spring.profiles.active=prod"]
---> Running in 22c055395746
Removing intermediate container 22c055395746
---> f2f3aed4d777

```

Déployer eureka-service :

```

PS C:\Users\AMINA\Desktop\Devoir 1\eureka-service> docker build -t eureka-service-web .
Sending build context to Docker daemon 46.81MB
Step 1/5 : FROM openjdk:17-alpine
---> 264c9bdce361
Step 2/5 : VOLUME /tmp
---> Running in 64ce4376c9c0
Removing intermediate container 64ce4376c9c0
---> fda419249d1d
---> 9b3639693163
---> Running in 58f767d180be

```

Déployer gateway-service :

```

PS C:\Users\AMINA\Desktop\Devoir 1\gateway-service> docker build -t gateway-service-web .
Sending build context to Docker daemon 42.79MB
Step 1/5 : FROM openjdk:17-alpine
---> 264c9bdce361
Step 2/5 : VOLUME /tmp
---> Using cache
Step 3/5 : COPY target/gateway-service-0.0.1-SNAPSHOT.jar /gateway.jar
---> f65ceccfde20
Step 4/5 : CMD ["java","-jar","/gateway.jar","--spring.profiles.active=prod"]
---> Running in 6071932cb536
Removing intermediate container 6071932cb536

```

Créer un nouveau fichier docker-compose.yml pour exécuter, à la fois, les image des micro services :

```
C:\Users\AMINA\Desktop\Devoir 1>docker-compose up -d
WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All containers will be scheduled on the current node.
To deploy your application across the swarm, use `docker stack deploy`.

Creating network "devoir1_default" with the default driver
Creating devoir1_eureka-service_1 ... done
Creating devoir1_gateway-service_1 ... done
Creating devoir1_customer-service_1 ... done
Creating devoir1_billing-service_1 ... done

C:\Users\AMINA\Desktop\Devoir 1>docker-compose ps

```

Name	Command	State	Ports
devoir1_billing-service_1	java -jar /billing-service ...	Up	0.0.0.0:8083->8083/tcp,:::8083->8083/tcp
devoir1_customer-service_1	java -jar /app.jar --sprin ...	Up	0.0.0.0:8082->8082/tcp,:::8082->8082/tcp
devoir1_eureka-service_1	java -jar /eureka-service. ...	Up	0.0.0.0:8761->8761/tcp,:::8761->8761/tcp
devoir1_gateway-service_1	java -jar /gateway.jar --s ...	Up	0.0.0.0:9999->9999/tcp,:::9999->9999/tcp

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
BILLING-SERVICE	n/a (1)	(1)	UP (1) - 862b013aad8:BILLING-SERVICE:8083
CUSTOMER-SERVICE	n/a (1)	(1)	UP (1) - 0739b60f0c3d:CUSTOMER-SERVICE:8082
GATEWAY	n/a (1)	(1)	UP (1) - afc5ddb24942:GATEWAY:9999

Swagger UI

Servers

http://192.168.43.200:8082 - Generated server url

customer-rest-api

GET /api/customers

POST /api/customers

GET /api/customers/{id}

Schemas

CustomerRequestDTO >

CustomerResponseDTO >

Non sécurisé | 192.168.43.200:8083/swagger-ui/index.html#/invoice-rest-controller

Windows Freecell | Anglais | Cours | COVID-19 Data Nor... | thohan88/covid19-... | WordPress Themes... | Bouffe - Restaurant... | Menu Simple - Bou... | jairidriss/Restauran...

GET /api/invoices

POST /api/invoices

GET /api/invoicesByCustomer/{customerID}

GET /api/invoices/{id}

Schemas

InvoiceRequestDTO >

Customer >

InvoiceResponseDTO >

Non sécurisé | 192.168.43.200:9999/CUSTOMER-SERVICE/api/customers

Windows Freecell | Anglais | Cours | COVID-19 Data Nor... | thohan88/covid19-...

```
1  [
2
3    {
4      "id": "A01",
5      "name": "Amina",
6      "email": "amina@gmail.com"
7    },
8
9    {
10     "id": "A02",
11     "name": "Fatima",
12     "email": "fatima@gmail.com"
13   }
14 ]
```