

Micro services project JEE Spring boot & Flutter



Réaliser par : Kaabi Mohamed et Laabid Haitem

Encadrer par : Mr Ayoub kebaili

5IIR EMSI Centre G2

INTRODUCTION

Dans ce projet on a souhaité de créer une application web permettant la communication entre les micro-services avec la notion de JWT. Ceci a permis de rentrer dans le détail de plusieurs concepts et d'améliorer et de comprendre des autres.

Le but de ce projet consiste en l'élaboration d'une application micro services web permettant la gestion des produits.

L'application a été modélisée tel que nous l'avons comprise et compte tenu des moyens matériels et temporels dont nous disposons. Mots clé : Spring, Spring Data, JPA, Hibernate, Spring CLOUD, Eureka, Spring Security et Json Web Token.

L'ARCHITECTURE GENERALE

C'est l'architecteur sur laquelle on va enchaîner la création de notre projet qui se compose de 5 micro-services :

Project <input checked="" type="radio"/> Maven Project <input type="radio"/> Gradle Project Spring Boot <input type="radio"/> 2.5.0 (SNAPSHOT) <input type="radio"/> 2.4.2 (SNAPSHOT) <input checked="" type="radio"/> 2.4.1 <input type="radio"/> 2.3.8 (SNAPSHOT) <input type="radio"/> 2.3.7 Project Metadata Group <input type="text" value="org.sid"/> Artifact <input type="text" value="customer-service"/> Name <input type="text" value="customer-service"/> Description <input type="text" value="Demo project for Spring Boot"/> Package name <input type="text" value="org.sid.customer-service"/> Packaging <input checked="" type="radio"/> Jar <input type="radio"/> War Java <input type="radio"/> 15 <input type="radio"/> 11 <input checked="" type="radio"/> 8	Language <input checked="" type="radio"/> Java <input type="radio"/> Kotlin <input type="radio"/> Groovy	Spring Web WEB Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.	Spring Data JPA SQL Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.	H2 Database SQL Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.	Eureka Discovery Client SPRING CLOUD DISCOVERY a REST based service for locating services for the purpose of load balancing and failover of middle-tier servers.	Rest Repositories WEB Exposing Spring Data repositories over REST via Spring Data REST.	Spring Boot DevTools DEVELOPER TOOLS Provides fast application restarts, LiveReload, and configurations for enhanced development experience.	Lombok DEVELOPER TOOLS Java annotation library which helps to reduce boilerplate code.	Spring Boot Actuator OPS Supports built in (or custom) endpoints that let you monitor and manage your application - such as application health, metrics, sessions, etc.
--	--	--	--	---	---	--	---	---	--

Project

Maven Project Gradle Project Java Kotlin Groovy

Spring Boot

2.5.0 (SNAPSHOT) 2.4.2 (SNAPSHOT) 2.4.1 2.3.8 (SNAPSHOT) 2.3.7

Project Metadata

Group: org.sid

Artifact: inventory-service

Name: inventory-service

Description: Demo project for Spring Boot

Package name: org.sid.inventory-service

Packaging: Jar War

Java: 15 11 8

Spring Web WEB

Build web, including RESTful applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Data JPA SQL

Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

H2 Database SQL

Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.

Eureka Discovery Client SPRING CLOUD DISCOVERY

a REST based service for locating services for the purpose of load balancing and failover of middle-tier servers.

Rest Repositories WEB

Exposing Spring Data repositories over REST via Spring Data REST.

Spring Boot DevTools DEVELOPER TOOLS

Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Lombok DEVELOPER TOOLS

Java annotation library which helps to reduce boilerplate code.

Spring Boot Actuator OPS

Supports built-in or custom endpoints that let you monitor and manage your application - such as application health, metrics, sessions, etc.



spring initializr

+

-

X

?

 **spring initializr**

Project Maven Project Gradle Project **Language** Java Kotlin Groovy

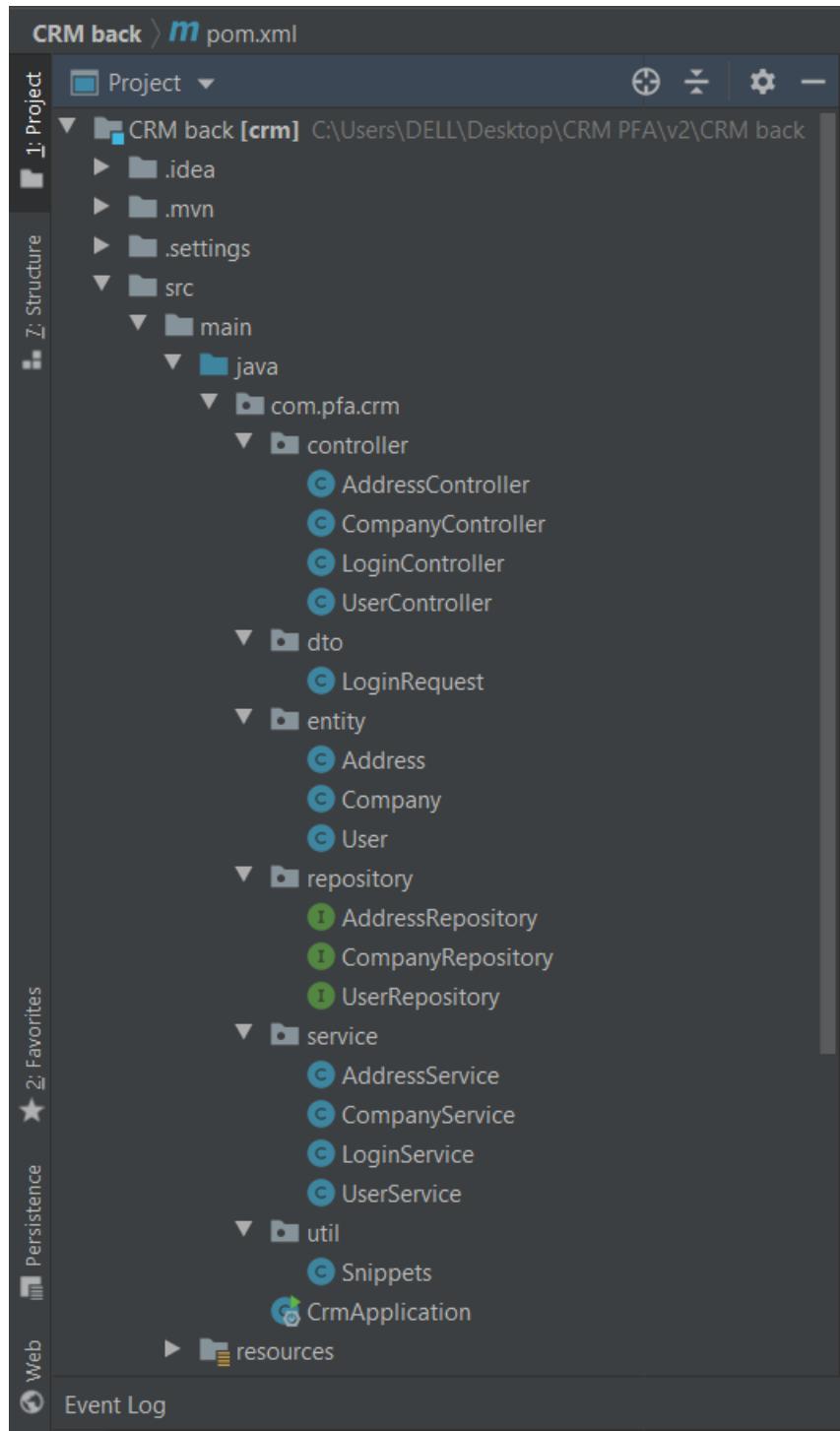
Spring Boot
 2.5.0 (SNAPSHOT) 2.4.2 (SNAPSHOT) 2.4.1
 2.3.8 (SNAPSHOT) 2.3.7

Project Metadata

Group	org.sid
Artifact	eureka-discovery
Name	eureka-discovery
Description	Demo project for Spring Boot
Package name	org.sid.eureka-discovery
Packaging	<input checked="" type="radio"/> Jar <input type="radio"/> War
Java	<input type="radio"/> 15 <input type="radio"/> 11 <input checked="" type="radio"/> 8

Dependencies ADD DEPENDENCIES... CTRL + B

Eureka Server SPRING CLOUD DISCOVERY
spring-cloud-netflix Eureka Server.



Project

- Maven Project
- Gradle Project
- Language
 - Java
 - Kotlin
 - Groovy

Spring Boot

- 2.5.0 (SNAPSHOT)
- 2.4.2 (SNAPSHOT)
- 2.4.1 (SNAPSHOT) **2.4.1**
- 2.3.8 (SNAPSHOT)
- 2.3.7

Project Metadata

- Group: org.sid
- Artifact: gateway-service
- Name: gateway-service
- Description: Demo project for Spring Boot
- Package name: org.sid.gateway-service
- Packaging: Jar
- Java: 15 11 8

Dependencies

- Eureka Discovery Client **SPRING CLOUD DISCOVERY**
a REST based service for locating services for the purpose of load balancing and failover of middle-tier servers.
- Spring Boot Actuator **OPS**
Supports built in or custom endpoints that let you monitor and manage your application - such as application health, metrics, sessions, etc.
- Gateway **SPRING CLOUD ROUTING**
Provides a simple yet effective way to route to APIs and provide cross cutting concerns to them such as security, monitoring/metrics, and resiliency.

ADD DEPENDENCIES... CTRL + B

inventory-service C:\Users\DELL\Desktop\TP-JEE\inventory-service

- .idea
- .mvn
- src
 - main
 - java
 - org.sid.inventoryservice
 - InventoryServiceApplication.java
 - InventoryServiceApplication
 - Product
 - ProductRepository
 - resources
 - application.properties
 - test
- target
 - .gitignore
 - inventory-service.iml
 - mvnw
 - mvnw.cmd
 - pom.xml
- External Libraries
- Scratches and Consoles

auth-service [sec-service] C:\Users\DELL\Desktop\TP-JEE\auth-service

- .idea
- .mvn
- src
 - main
 - java
 - org.sid.secservice
 - sec
 - entities
 - AppRole
 - AppUser
 - filter
 - JwtAuthenticationFiltre
 - JwtAuthorizationFilter
 - repositories
 - AppRoleRepository
 - AppUserRepository
 - service
 - AccountService
 - AccountServiceImpl
 - UserDetailsServiceimpl
 - web
 - AcoutRestController
 - RoleUserForm
 - JWTUtil
 - SecurityConfig
 - SecServiceApplication
 - resources
 - application.properties
 - target
 - .gitignore
 - mvnw
 - mvnw.cmd

```

@FeignClient(name = "CUSTOMER-SERVICE")
public interface CustomerRestClient {
    @GetMapping(path="/customers/{id}")
    Customer getCustomerById(@PathVariable(name="id") Long id);
}

```

```

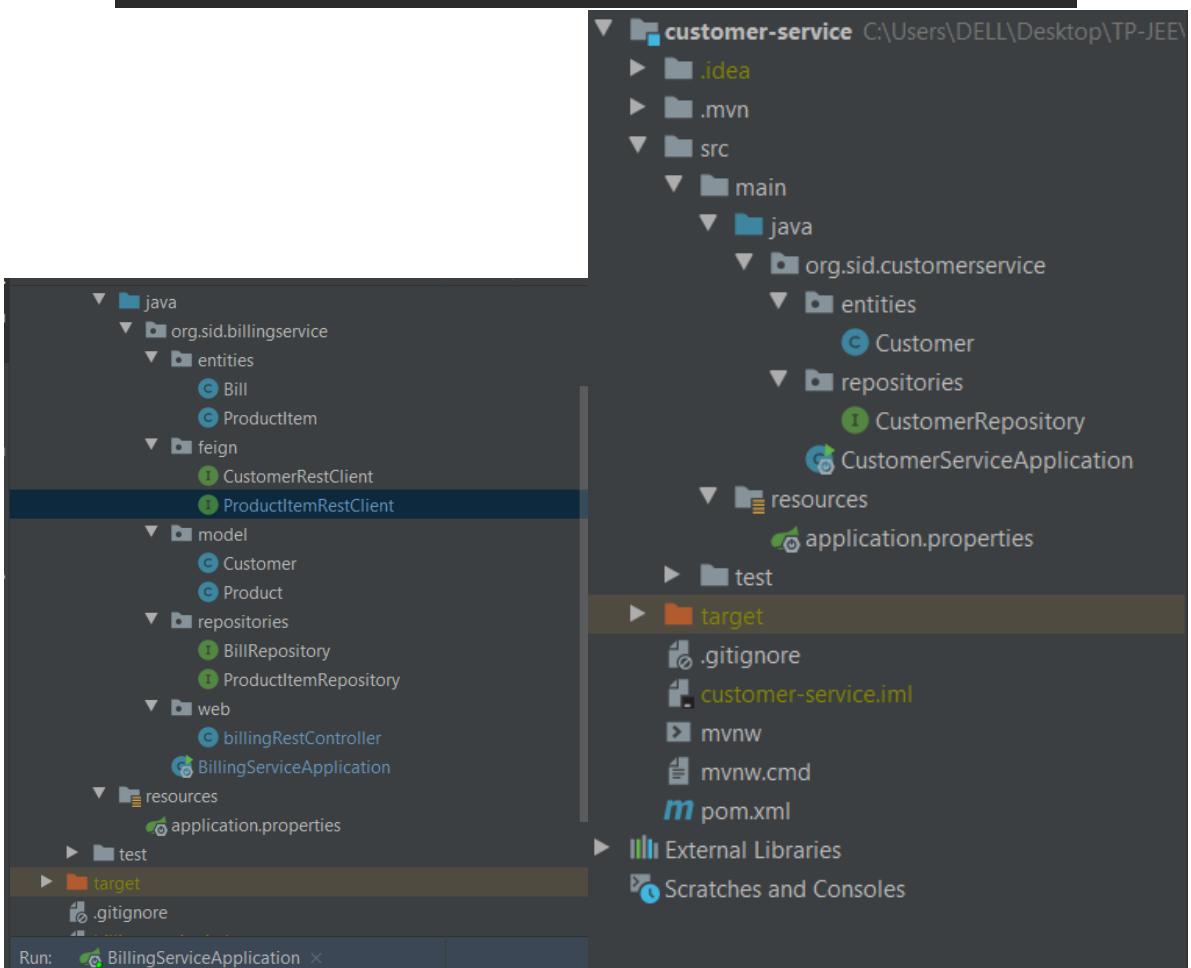
#management.endpoints.web.exposure.include=*
server.port=8081
spring.application.name=customer-service
spring.datasource.url=jdbc:h2:mem:customer-db
spring.cloud.discovery.enabled=true

@FeignClient(name = "PRODUCTS-SERVICE")

public interface ProductItemRestClient {
    @GetMapping(path = "/products")
    PagedModel<Product>pageProducts();
    @GetMapping(path = "/products/{id}")
    Product getProductById(@PathVariable Long id);
}

server.port=8033
spring.application.name=products-service
spring.datasource.url=jdbc:h2:mem:products-db
spring.cloud.discovery.enabled=true
#management.endpoints.web.exposure.include=*

```



File Structure Comparison:

```

eureka-discovery C:\Users\DELL\Desktop\TP-JEE\eureka-discovery
  - .idea
  - .mvn
  - src
    - main
      - java
        - org.sid.eurekadiscovery
          - EurekaDiscoveryApplication
    - resources
      - application.properties
  - test
  - target
    - .gitignore
    - eureka-discovery.iml
    - mvnw
    - mvnw.cmd
    - pom.xml
  - External Libraries
  - Scratches and Consoles

gateway C:\Users\DELL\Desktop\TP-JEE\gateway
  - .idea
  - .mvn
  - src
    - main
      - java
        - org.sid.gateway
          - GatewayApplication
    - resources
      - application.properties
      - application.yml
    - test
  - target
    - .gitignore
    - gateway.iml
    - mvnw
    - mvnw.cmd
    - pom.xml
  - External Libraries
  - Scratches and Consoles

```

Code Editor View:

```

1 package com.pfa.crm.controller;
2
3 import ...
4
5 @CrossOrigin
6 @RestController
7 @RequestMapping("/api/companies")
8 public class CompanyController {
9
10
11     @Autowired
12     private CompanyService companyService;
13
14     // COMPANIES
15
16     // getCompanies
17     @GetMapping("")
18     public List<Company> getCompanies() { return this.companyService.getCompanies(); }
19
20     // getCompanyById
21     @GetMapping("/{companyId}")
22     public Optional<Company> getCompanyById(@PathVariable Long companyId) {
23         return this.companyService.getCompanyById(companyId);
24     }
25
26     // searchCompaniesByReferenceOrName
27     @GetMapping("/search")
28     public List<Company> searchCompaniesByReferenceOrName(@RequestParam String term) {
29         return this.companyService.searchCompaniesByReferenceOrName(term);
30     }
31
32 }

```

```
52     // createCompany
53     @PostMapping("/{userId}")
54     public Company createCompany(@PathVariable Long userId, @RequestBody Company company) {
55         return this.companyService.createCompany(userId, company);
56     }
57
58     // updateCompany
59     @PatchMapping("")
60     public Company updateCompany(@RequestBody Company company) { return this.companyService.updateCompany(company); }
61
62     // deleteCompany
63     @DeleteMapping("/{companyId}")
64     public void deleteCompany(@PathVariable Long companyId) { this.companyService.deleteCompany(companyId); }
65
66     // USERS COMPANIES
67
68     // getUsersByCompanyId
69     @GetMapping("{companyId}/users")
70     public Set<User> getUsersByCompanyId(@PathVariable Long companyId) {
71         return this.companyService.getUsersByCompanyId(companyId);
72     }
73
74     // addUserToCompany
75     @PostMapping("{companyId}/users/{userId}")
76     public User addUserToCompany(@PathVariable Long companyId, @PathVariable Long userId) {
77         return this.companyService.addUserToCompany(companyId, userId);
78     }
79
80     // UserController
81     @CrossOrigin
82     @RestController
83     @RequestMapping("/api/users")
84     public class UserController {
85
86         @Autowired
87         private UserService userService;
88
89         // USERS
90
91         // getUsers
92         @GetMapping("")
93         public List<User> getUsers() { return this.userService.getUsers(); }
94
95         // getUserId
96         @GetMapping("/{userId}")
97         public Optional<User> getUserId(@PathVariable Long userId) { return this.userService.getUserById(userId); }
98
99         // searchUsersByName
100        @GetMapping("/search")
101        public List<User> searchUsersByName(@RequestParam String term) { return this.userService.searchUsersByName(term); }
102
103        // createUser
104        @PostMapping("")
105        public User createUser(@RequestBody User user) { return this.userService.createUser(user); }
106
107        // updateUser
108        @PatchMapping("")
109        public User updateUser(@RequestBody User user) { return this.userService.updateUser(user); }
110    }
111
112    // updateUserAddress
113    @PatchMapping("/{userId}/updateaddress")
114    public User updateUserAddress(@PathVariable Long userId, @RequestBody Address address) {
115        return this.userService.updateUserAddress(userId, address);
116    }
117}
```

```
63     // deleteUser
64     @DeleteMapping("/{userId}")
65     public void deleteUser(@PathVariable Long userId) { this.userService.deleteUser(userId); }
66
67     // USERS COMPANIES
68
69     // getCompaniesByUserId
70     @GetMapping("/{userId}/companies")
71     public Set<Company> getCompaniesByUserId(@PathVariable Long userId) {
72         return this.userService.getCompaniesByUserId(userId);
73     }
74
75     // addCompanyToUser
76     @PostMapping("{userId}/companies/{companyId}")
77     public Company addCompanyToUser(@PathVariable Long userId, @PathVariable Long companyId){
78         return this.userService.addCompanyToUser(userId, companyId);
79     }
80
81     // removeCompanyFromUser
82     @DeleteMapping("/{userId}/companies/{companyId}")
83     public Company removeCompanyFromUser(@PathVariable Long userId, @PathVariable Long companyId) {
84         return this.userService.removeCompanyFromUser(userId, companyId);
85     }
86
87     // ADDRESSES
88
89     // createUserAddress
90     @PostMapping("/{userId}/createaddress")
91     public User createUserAddress(@PathVariable Long userId, @RequestBody Address address) {
92         return this.userService.createUserAddress(userId, address);
93     }
94
95     package com.pfa.crm.controller;
96
97     import ...
98
99     @CrossOrigin
100    @RestController
101    @RequestMapping("/api/login")
102    public class LoginController {
103
104        @Autowired
105        private LoginService loginService;
106
107        @PostMapping("")
108        public Optional<User> findByEmailAndPassword(@RequestBody LoginRequest loginRequest) {
109            return this.loginService.findByEmailAndPassword(loginRequest);
110        }
111    }
112 }
```

```
package com.pfa.crm.entity;

import ...


@Entity
@Table(name = "Addresses")
public class Address implements Serializable {

    private static final long serialVersionUID = -5508310782410021980L;

    @Column
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    @Column(length = 5)
    private int streetNumber;

    @Column(length = 30)
    private String streetName;

    @Column(length = 20)
    private String city;

    @Column(length = 10)
    private String postcode;

    @Column(length = 20)
    private String region;
```

```
21  @Entity
22  @Table(name = "Companies")
23  public class Company implements Serializable {
24
25      private static final long serialVersionUID = -7491177392L;
26
27      @Column
28      @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
29      private long id;
30
31      @Column(length = 10)
32      private String reference;
33
34      @Column(length = 30)
35      private String name;
36
37      @Column(length = 30)
38      private String owner;
39
40      @Column(length = 20)
41      private String telephone;
42
43      @Column(length = 50)
44      private String website;
45
46      @Column(length = 30)
47      private String contactName;
48
49      @Column(length = 20)
50      private String contactTelephone;
```

```
51  
52     @Column(nullable = false, unique = true)  
53     private String contactEmail;  
54  
55     @Column  
56     private Date callLast;  
57  
58     @Column  
59     private Date visitLast;  
60  
61     @Column(length = 555)  
62     private String messages;  
63  
64     @Column(length = 30)  
65     private String discount;  
66  
67     @Column(length = 50)  
68     private float salesExp;  
69  
70     @Column(length = 20)  
71     private float sales3m;  
72  
73     @Column(length = 20)  
74     private float salesTotal;  
75  
76     @OneToOne(cascade = CascadeType.ALL)  
77     private Address address;
```

```
78  
79     @ManyToMany(mappedBy = "companies")  
80     @JsonIgnore  
81     Set<User> users = new HashSet<>();  
82  
83     public Company() {}  
84  
85     @PreRemove  
86     public void removeUsers() {  
87         this.users.forEach((user) -> {  
88             user.removeCompany(this);  
89         });  
90     }  
91  
92     public void addUser(User user) {  
93         if (!this.users.contains(user)) {  
94             this.users.add(user);  
95         }  
96     }  
97  
98     public void removeUser(User user) {  
99         if (!this.users.contains(user)) {  
100            this.users.remove(user);  
101        }  
102    }
```

```
21  @Entity
22  @Table(name = "Users")
23  public class User implements Serializable {
24
25      private static final long serialVersionUID = -1711112695555661590L;
26
27      @Column
28      @Id
29      @GeneratedValue(strategy = GenerationType.IDENTITY)
30      private long id;
31
32      @Column(length = 20)
33      private String firstName;
34
35      @Column(length = 15)
36      private String lastName;
37
38      @Column(nullable = false, unique = true)
39      private String email;
40
41      @Column(nullable = false)
42      private String password;
43
44      @Column(length = 20)
45      private String telephone;
46
47      @Column(length = 30)
48      private String role;
49
```

```
49
50     @OneToOne(cascade = CascadeType.ALL)
51     private Address address;
52
53     @ManyToMany(fetch = FetchType.LAZY)
54     @JsonIgnore
55     Set<Company> companies = new HashSet<>();
56
57     public User() {
58
59     }
60
61     @PreRemove
62     public void removeCompanies() {
63         this.companies.forEach((company) -> {
64             company.removeUser(this);
65         });
66     }
67
68     public void addCompany(Company company) {
69         if (!this.companies.contains(company)) {
70             this.companies.add(company);
71         }
72     }
73
74     public void removeCompany(Company company) {
75         if (this.companies.contains(company)) {
76             this.companies.remove(company);
77         }
78     }
```

DTO : DTO, which stands for Data Transfer Object, is a design pattern conceived to reduce the number of calls when working with remote interfaces

```
package com.pfa.crm.dto;

public class LoginRequest {

    private String email;
    private String password;

    public LoginRequest() {}

    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }
    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }

}
```

```
1 package com.pfa.crm.controller;
2
3 import ...
16
17 @RestController
18 @RequestMapping("/api/addresses")
19 public class AddressController {
20
21     @Autowired
22     private AddressService addressService;
23
24     // ADDRESSES
25
26     // getAddresses
27     @GetMapping("")
28     public List<Address> getAddresses() { return this.addressService.getAddresses(); }
31
32     // getAddressById
33     @GetMapping("/{id}")
34     public Optional<Address> getAddressById(@PathVariable Long id) { return this.addressService.getAddressById(id); }
37
38     // createAddress
39     @PostMapping("")
40     public Address createAddress(@RequestBody Address address) { return this.addressService.createAddress(address); }
43
44     // updateAddress
45     @PatchMapping("/{id}")
46     public Address updateAddress(@RequestBody Address address) { return this.addressService.updateAddress(address); }
49
50 }
```

Repositories for CRM-service :

```

@Repository
public interface AddressRepository extends JpaRepository<Address, Long> {

}

@Repository
public interface CompanyRepository extends JpaRepository<Company, Long>{
    |
    List<Company> findByReferenceIgnoreCaseContainingOrNameIgnoreCaseContaining(String reference, String name);
}

@Repository
public interface UserRepository extends JpaRepository<User, Long>{
    |
    Optional<User> findByEmailAndPassword(String email, String password);
    List<User> findByNameIgnoreCaseContainingOrLastNameIgnoreCaseContaining(String firstName, String lastName);
}

```

Services de CRM-service :

Entreprise :

```

1 package com.pfa.crm.service;
2
3 import ...
4
5
6 @Service
7 public class CompanyService {
8
9     @Autowired
10    private CompanyRepository companyRepository;
11
12    @Autowired
13    private UserRepository userRepository;
14
15    // COMPANIES
16
17    // getCompanies
18    public List<Company> getCompanies() { return this.companyRepository.findAll(); }
19
20    // getCompanyById
21    public Optional<Company> getCompanyById(Long companyId) { return this.companyRepository.findById(companyId); }
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37

```

```

38
39     // searchCompaniesByReferenceOrName
40     public List<Company> searchCompaniesByReferenceOrName(String term) {
41         return this.companyRepository.findByReferenceIgnoreCaseContainingOrNameIgnoreCaseContaining(term, term);
42     }
43
44     // createCompany
45     public Company createCompany(Long userId, Company company) {
46         User user = this.userRepository.findById(userId).orElseThrow(() -> new RuntimeException("User not found"));
47         Company newCompany = this.companyRepository.save(company);
48         user.addCompany(newCompany);
49         this.userRepository.save(user);
50         return newCompany;
51     }
52
53     // updateCompany
54     public Company updateCompany(Company company) { return this.companyRepository.save(company); }
55
56     // deleteCompany
57     public void deleteCompany(Long companyId) { this.companyRepository.deleteById(companyId); }
58
59
60
61     // USERS
62
63
64     // getUsersByCompanyId
65     public Set<User> getUsersByCompanyId(Long companyId) {
66         Company company = this.getCompanyById(companyId).orElseThrow(() -> new RuntimeException("Company not found"));
67         return company.getUsers();
68     }
69
70
71     // addUserToCompany
72     public User addUserToCompany(Long companyId, Long userId) {
73         Company company = this.getCompanyById(companyId).orElseThrow(() -> new RuntimeException("Company not found"));
74         User user = this.userRepository.findById(userId).orElseThrow(() -> new RuntimeException("User not found"));
75         user.addCompany(company);
76         return this.userRepository.save(user);
77     }
78
79
80     // removeUserFromCompany
81     public User removeUserFromCompany(Long companyId, Long userId) {
82         Company company = this.getCompanyById(companyId).orElseThrow(() -> new RuntimeException("Company not found"));
83         User user = this.userRepository.findById(userId).orElseThrow(() -> new RuntimeException("User not found"));
84         user.removeCompany(company);
85         return this.userRepository.save(user);
86     }
87
88     // ADDRESSES
89
90     // createCompanyAddress
91     public Company createCompanyAddress(Long companyId, Address createCompanyAddress) {
92         Company company = this.getCompanyById(companyId).orElseThrow(() -> new RuntimeException("Company not found"));
93         company.setAddress(createCompanyAddress);
94         return this.updateCompany(company);
95     }
96
97     // updateCompanyAddress
98     public Company updateCompanyAddress(Long companyId, Address updateCompanyAddress) {
99         Company company = this.getCompanyById(companyId).orElseThrow(() -> new RuntimeException("Company not found"));
100        company.setAddress(updateCompanyAddress);
101        return this.updateCompany(company);
102    }

```

Addresse :

```
1 package com.pfa.crm.service;
2
3 import ...
11
12 @Service
13 public class AddressService {
14
15     @Autowired
16     private AddressRepository addressRepository;
17
18     // ADDRESSES
19
20     // getAddresses
21     public List<Address> getAddresses() { return this.addressRepository.findAll(); }
24
25     // getAddressById
26     public Optional<Address> getAddressById(Long id) { return this.addressRepository.findById(id); }
29
30     // createAddress
31     public Address createAddress(Address address) { return this.addressRepository.save(address); }
34
35     // updateAddress
36     public Address updateAddress(Address address) { return this.addressRepository.save(address); }
39 }
```

Login :

```
1 package com.pfa.crm.service;
2
3 import ...
11
12 @Service
13 public class LoginService {
14
15     @Autowired
16     private UserRepository userRepository;
17
18     @ ...
19     public Optional<User> findByEmailAndPassword(LoginRequest loginRequest) {
20         return this.userRepository.findByEmailAndPassword(loginRequest.getEmail(), loginRequest.getPassword());
21     }
22 }
23
```

User :

```
1 package com.pfa.crm.service;
2
3 import ...
13
14 @Service
15 public class UserService {
16
17     @Autowired
18     private UserRepository userRepository;
19     @Autowired
20     private CompanyService companyService;
21
22     // USERS
23     |
24     // getUsers
25     public List<User> getUsers() { return this.userRepository.findAll(); }
28
29     // getUserById
30     public Optional<User> getUserId(Long userId) { return this.userRepository.findById(userId); }
33
34     // searchUsersByName
35     public List<User> searchUsersByName(String term) {
36         return this.userRepository.findByFirstNameIgnoreCaseContainingOrLastNameIgnoreCaseContaining(term, term);
37     }
38 }
```

```
38
39     // createUser
40     public User createUser(User user) { return this.userRepository.save(user); }
41
42     // updateUser
43     public User updateUser(User user) { return this.userRepository.save(user); }
44
45     // deleteUser
46     public void deleteUser(Long userId) { this.userRepository.deleteById(userId); }
47
48
49
50
51
52
53
54
```

```
56     // COMPANIES
57     // getCompaniesByUserId
58     public Set<Company> getCompaniesByUserId(Long userId) {
59         User user = this.getUserId(userId).orElseThrow(() -> new RuntimeException("User not found"));
60         return user.getCompanies();
61     }
62
63     // addCompanyToUser
64     public Company addCompanyToUser(Long userId, Long companyId) {
65         User user = this.getUserId(userId).orElseThrow(() -> new RuntimeException("User not found"));
66         Company company = this.companyService.getCompanyById(companyId).orElseThrow(() -> new RuntimeException("Company not found"));
67         user.addCompany(company);
68         this.userRepository.save(user);
69         return company;
70     }
71
72     // removeCompanyFromUser
73     public Company removeCompanyFromUser(Long userId, Long companyId) {
74         User user = this.getUserId(userId).orElseThrow(() -> new RuntimeException("User not found"));
75         Company company = this.companyService.getCompanyById(companyId).orElseThrow(() -> new RuntimeException("Company not found"));
76         user.removeCompany(company);
77         this.userRepository.save(user);
78         return company;
79     }
```

```
82     // ADDRESSES
83
84     // createUserAddress
85     public User createUserAddress(Long userId, Address createUserAddress) {
86         User user = this.getUserId(userId).orElseThrow(() -> new RuntimeException("User not found"));
87         user.setAddress(createUserAddress);
88         return this.updateUser(user);
89     }
90
91     // updateUserAddress
92     public User updateUserAddress(Long userId, Address updateUserAddress) {
93         User user = this.getUserId(userId).orElseThrow(() -> new RuntimeException("User not found"));
94         user.setAddress(updateUserAddress);
95         return this.updateUser(user);
96     }
97
98
99     }
100 }
```

BASE DE DONNEE :

jdbc:h2:~/crmDB

- ADDRESSSES
- COMPANIES
- USERS
- USERS_COMPANIES
- INFORMATION_SCHEMA**
- Sequences
- Users

H2 1.4.200 (2019-10-14)

USERS

- ID
- EMAIL
- FIRST_NAME
- LAST_NAME
- PASSWORD
- ROLE
- TELEPHONE
- ADDRESS_ID
- Indexes

USERS_COMPANIES

- USERS_ID
- COMPANIES_ID
- Indexes

INFORMATION_SCHEMA

Sequences

Users

H2 1.4.200 (2019-10-14)

Auto commit | Max rows: 1000 | Run | Run Selected | Auto complete | Clear | SQL statement:

Important Commands

	Displays this Help Page
	Shows the Command History
	Ctrl+Enter Executes the current SQL statement
	Shift+Enter Executes the SQL statement defined by the text selection
	Ctrl+Space Auto complete
	Disconnects from the database

Sample SQL Script

Delete the table if it exists	DROP TABLE IF EXISTS TEST;
Create a new table	CREATE TABLE TEST(ID INT PRIMARY KEY, NAME VARCHAR(255));
with ID and NAME columns	
Add a new row	INSERT INTO TEST VALUES(1, 'Hello');
Add another row	INSERT INTO TEST VALUES(2, 'World');
Query the table	SELECT * FROM TEST ORDER BY ID;
Change data in a row	UPDATE TEST SET NAME='Hi' WHERE ID=1;
Remove a row	DELETE FROM TEST WHERE ID=2;
Help	HELP ...

Adding Database Drivers

Additional database drivers can be registered by adding the Jar file location of the driver to the environment variable H2DRIVERS to C:/Programs/hsqldb/lib/hsqldb.jar.

Costumer-service :

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "costumerSpringCloud [costumermicroservice]". It contains a "src" directory with "main" and "test" packages. "main" has "java" and "resources" sub-directories. "resources" contains "application.properties".
- Code Editor:** The file "CostumermicroServiceSpringCloudandEurekaRegistryImplementationApplication.java" is open. The code defines a main method that starts a Spring Application and saves three Customer objects to a repository.
- Console:** The console output shows log messages indicating the saving of three customers with names "mohammed", "hassan", and "abdo" to the database.
- Status Bar:** The status bar shows the current time as 12:14, file encoding as CRLF, and the current branch as master.

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "costumerSpringCloud [costumermicroservice]". It contains a "src" directory with "main" and "test" packages. "main" has "java" and "resources" sub-directories. "resources" contains "application.properties".
- Code Editor:** The file "application.properties" is open, showing configuration properties for the service.
- Console:** The console output shows log messages indicating the resolution of Eureka endpoints via configuration.
- Status Bar:** The status bar shows the current time as 12:14, file encoding as CRLF, and the current branch as master.

The screenshot shows the IntelliJ IDEA interface with the Customer.java file open in the editor. The code defines a Customer entity with an ID, name, and email. The project structure on the left shows the com.costumer.costumermicroservice package containing the Customer entity and its repository.

```
package com.costumer.costumermicroservice.entities;  
import ...  
@Entity  
@Data @AllArgsConstructor @NoArgsConstructor @ToString  
public class Customer {  
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Long id;  
    private String name;  
    private String email;  
}
```

The screenshot shows the IntelliJ IDEA interface with the CustomerRepository.java code open in the editor. It defines a JPA repository interface for the Customer entity. The project structure on the left shows the com.costumer.costumermicroservice.repository package containing the CustomerRepository interface.

```
package com.costumer.costumermicroservice.repository;  
import ...  
@RepositoryRestResource  
public interface CustomerRepository extends JpaRepository<Customer, Long> {  
}
```

```

{
  "_embedded": {
    "customers": [
      {
        "id": 1,
        "name": "mohammed",
        "email": "mohammed@gmail.com",
        "_links": {
          "self": {
            "href": "http://10.10.1.81:8081/customers/1"
          },
          "customer": {
            "href": "http://10.10.1.81:8081/customers/1"
          }
        }
      },
      {
        "id": 2,
        "name": "hassan",
        "email": "hassan@gmail.com",
        "_links": {
          "self": {
            "href": "http://10.10.1.81:8081/customers/2"
          },
          "customer": {
            "href": "http://10.10.1.81:8081/customers/2"
          }
        }
      },
      {
        "id": 3,
        "name": "abdo",
        "email": "abdo@gmail.com",
        "_links": {
          "self": {
            "href": "http://10.10.1.81:8081/customers/3"
          }
        }
      }
    ]
  }
}

```

Product-service :

```

package com.inventoryservice.inventoryservice;
import ...
@SpringBootApplication
public class InventoryserviceApplication {
    public static void main(String[] args) {
        SpringApplication.run(InventoryserviceApplication.class, args);
    }
    @Bean
    CommandLineRunner start(ProductRepository productRepository, RepositoryRestConfiguration restConfiguration) {
        restConfiguration.exposeIdsFor(Products.class);
        return args -> {
            productRepository.save(new Product(id: null, name: "ordin", price: 456, quantity: 78));
            productRepository.save(new Product(id: null, name: "telef", price: 786, quantity: 214));
            productRepository.save(new Product(id: null, name: "clavier", price: 234, quantity: 444));
            productRepository.findAll().forEach(p-> System.out.println(p.toString()));
        };
    }
    @Entity
    @Data @AllArgsConstructor @NoArgsConstructor @ToString
}

```

The screenshot shows the IntelliJ IDEA interface with the code editor open to `InventoryserviceApplication.java`. The code defines a `Product` entity with fields `id`, `name`, `price`, and `quantity`. It also defines a `ProductRepository` interface extending `JpaRepository<Product, Long>`. The code uses annotations like `@Entity`, `@Data`, `@Id`, and `@GeneratedValue(strategy = GenerationType.IDENTITY)`.

```
InventoryserviceApplication.java
...
@Entity
@Data @AllArgsConstructor @NoArgsConstructor @ToString
class Product{
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private double price;
    private double quantity;
}
@RepositoryRestResource
interface ProductRepository extends JpaRepository<Product, Long>{}
```

The screenshot shows a browser window displaying a JSON response from the URL `localhost:8888/PRODUCT-SERVICE/products`. The response is a list of products, each with an ID, name, price, quantity, and two links: `self` and `product`.

```
{
  "_embedded": {
    "products": [
      {
        "id": 1,
        "name": "ordin",
        "price": 456,
        "quantity": 78,
        "_links": {
          "self": {
            "href": "http://10.10.1.81:8082/products/1"
          },
          "product": {
            "href": "http://10.10.1.81:8082/products/1"
          }
        }
      },
      {
        "id": 2,
        "name": "telef",
        "price": 786,
        "quantity": 214,
        "_links": {
          "self": {
            "href": "http://10.10.1.81:8082/products/2"
          },
          "product": {
            "href": "http://10.10.1.81:8082/products/2"
          }
        }
      },
      {
        "id": 3,
        "name": "clavier",
        "price": 234,
        "quantity": 444
      }
    ]
  }
}
```

Gitway :

The screenshot shows the IntelliJ IDEA interface with the project 'gitway-service' open. The code editor displays the `GitwayServiceApplication.java` file, which contains the main application class. The code defines a `@SpringBootApplication` annotated class with a `main` method and a `routeLocator` configuration method. The `application.properties` file is also visible in the project structure.

```
package com.gatewayservice.gatewayservice;
import ...
@SpringBootApplication
public class GitwayServiceApplication {
    public static void main(String[] args) { SpringApplication.run(GitwayServiceApplication.class, args);
    }
    @Bean
    RouteLocator routeLocator(RouteLocatorBuilder builder){
        return builder.routes()
            .route((r)->r.path("/customers/**").uri("lb://CUSTOMER-SERVICE"))
            .route((r)->r.path("/products/**").uri("lb://PRODUCT-SERVICE"))
            .build();
    }
    @Bean
    DiscoveryClientRouteDefinitionLocator definitionLocator(ReactiveDiscoveryClient rdc, DiscoveryLocator
        return new DiscoveryClientRouteDefinitionLocator(rdc,properties);
}
}
```

The screenshot shows the IntelliJ IDEA interface with the project 'gitway-service' open. The code editor displays the `application.properties` file, which contains the following configuration properties:

```
server.port=8888
spring.application.name=gateway-service
spring.cloud.discovery.enabled=true
eureka.instance.preferIpAddress=true
```

Billing-service :

The screenshot shows the IntelliJ IDEA interface with the project 'billing-service' open. The code editor displays the `Bill.java` file under the package `org.sid.billingservice.entities`. The code defines a `Bill` class with fields for `id`, `billingDate`, `productItems`, `customerID`, and `customer`. Annotations include `@Entity`, `@Data`, `@Id @GeneratedValue(strategy = GenerationType.IDENTITY)`, and `@Transient`.

```
1 package org.sid.billingservice.entities;
2
3 import ...
4
5 @Entity
6 @Data
7 @AllArgsConstructor
8 @NoArgsConstructor
9 @ToString
10
11 public class Bill {
12     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
13     private Long id;
14     private Date billingDate;
15     @OneToMany(mappedBy = "bill")
16     private Collection<ProductItem> productItems;
17     private Long customerID;
18     @Transient
19     private Customer customer;
20 }
```

The screenshot shows the IntelliJ IDEA interface with the project 'billing-service' open. The code editor displays the `ProductItem.java` file under the package `org.sid.billingservice.entities`. The code defines a `ProductItem` class with fields for `id`, `quantity`, `price`, `productID`, `product`, and `productName`. Annotations include `@Entity`, `@Data`, `@AllArgsConstructor`, `@NoArgsConstructor`, `@ToString`, `@Id @GeneratedValue(strategy = GenerationType.IDENTITY)`, `@ManyToOne`, `@Transient`, and `@JsonProperty(access = JsonProperty.Access.WRITE_ONLY)`.

```
1 package org.sid.billingservice.entities;
2
3 import ...
4
5 @Entity
6 @Data
7 @AllArgsConstructor
8 @NoArgsConstructor
9 @ToString
10
11 public class ProductItem {
12     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
13     private Long id;
14     private double quantity;
15     private double price;
16     private Long productID;
17     @ManyToOne
18     private Bill bill;
19
20     @Transient
21     private Product product;
22     @Transient
23     private String productName;
24 }
```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help billing-service [src] main java org.sid.billingservice feign CustomerRestClient

```
package org.sid.billingservice.feign;
import ...;

@FeignClient(name = "CUSTOMER-SERVICE")
public interface CustomerRestClient {
    @GetMapping(path = "/customers/{id}")
    public Customer getCustomerById(@PathVariable(name = "id") Long id);
}
```

Capture Plein écran

Project: billing-service C:\Users\user\Desktop\Nouveau dossier
Main.java ProductItem.java CustomerRestClient.java
src main java org.sid.billingservice feign CustomerRestClient ProductItemRestClient
model repository web BillingServiceApplication
resources test target
gitignore billing-service.iml mvnw mvnw.cmd pom.xml
External Libraries Scratches and Consoles

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help billing-service [src] main java org.sid.billingservice feign ProductItemRestClient

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help billing-service [src] main java org.sid.billingservice feign ProductItem.java CustomerRestClient.java ProductItemRestClient.java
```

```
package org.sid.billingservice.feign;
import ...;

@FeignClient(name = "PRODUCT-SERVICE")
public interface ProductItemRestClient {
    @GetMapping(path = "/products")
    PagedModel<Product> pageProducts();

    @GetMapping(path = "/products/{id}")
    Product getProductById(@PathVariable Long id);
}
```

Project: billing-service C:\Users\user\Desktop\Nouveau dossier
Main.java ProductItem.java CustomerRestClient.java ProductItemRestClient.java
src main java org.sid.billingservice feign CustomerRestClient ProductItemRestClient
model repository web BillingServiceApplication
resources test target
gitignore billing-service.iml mvnw mvnw.cmd pom.xml
External Libraries Scratches and Consoles

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help billing-service [src] main java org.sid.billingservice feign ProductItemRestClient

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help billing-service [src] main java org.sid.billingservice feign ProductItem.java CustomerRestClient.java ProductItemRestClient.java
```

```
package org.sid.billingservice.feign;
import ...;

@FeignClient(name = "PRODUCT-SERVICE")
public interface ProductItemRestClient {
    @GetMapping(path = "/products")
    PagedModel<Product> pageProducts();

    @GetMapping(path = "/products/{id}")
    Product getProductById(@PathVariable Long id);
}
```

Project: billing-service C:\Users\user\Desktop\Nouveau dossier
Main.java ProductItem.java CustomerRestClient.java ProductItemRestClient.java
src main java org.sid.billingservice feign CustomerRestClient ProductItemRestClient
model repository web BillingServiceApplication
resources test target
gitignore billing-service.iml mvnw mvnw.cmd pom.xml
External Libraries Scratches and Consoles

The screenshot shows the IntelliJ IDEA interface with the project 'billing-service' open. The code editor displays the `Customer.java` file:

```
package org.sid.billingservice.model;

import lombok.Data;

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

The code editor has tabs for `Bill.java`, `ProductItem.java`, `CustomerRestClient.java`, `ProductItemRestClient.java`, and `Customer.java`. The bottom status bar shows 'Event Log' and the current time as 6:14.

The screenshot shows the IntelliJ IDEA interface with the project 'billing-service' open. The code editor displays the `Product.java` file:

```
package org.sid.billingservice.model;

import lombok.Data;

@Data
public class Product {
    private Long id;
    private String name;
    private double price;
    private double quantity;
}
```

The code editor has tabs for `Bill.java`, `ProductItem.java`, `CustomerRestClient.java`, `ProductItemRestClient.java`, `Customer.java`, and `Product.java`. The bottom status bar shows 'Event Log' and the current time as 6:14.

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

billing-service [src] main java org.sid.billingservice.repository BillRepository

ProductItem.java CustomerRestClient.java ProductItemRestClient.java Customer.java Product.java BillRepository.java

Project

billing-service C:\Users\user\Desktop\Nouveau dossier

src

main

java

org.sid.billingservice

entities

feign

model

repository

BillRepository

ProductItemRepository

web

BillingServiceApplication

resources

test

target

gitignore

billing-service.iml

mvnw

mvnw.cmd

pom.xml

External Libraries

Scratches and Consoles

1 package org.sid.billingservice.repository;

2 import ...;

3

4

5

6

7

8 public interface BillRepository extends JpaRepository<Bill, Long> {

9 }

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756

757

758

759

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

928

929

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000

1001

1002

1003

1004

1005

1006

1007

1008

1009

1010

1011

1012

1013

1014

1015

1016

1017

1018

1019

1020

1021

1022

1023

1024

1025

1026

1027

1028

1029

1030

1031

1032

1033

1034

1035

1036

1037

1038

1039

1040

1041

1042

1043

1044

1045

1046

1047

1048

1049

1050

1051

1052

1053

1054

1055

1056

1057

1058

1059

1060

1061

1062

1063

1064

1065

1066

1067

1068

1069

1070

1071

1072

1073

1074

1075

1076

1077

1078

1079

1080

1081

1082

1083

1084

1085

1086

1087

1088

1089

1090

1091

1092

1093

1094

1095

1096

1097

1098

1099

1100

1101

1102

1103

1104

1105

1106

1107

1108

1109

1110

1111

1112

1113

1114

1115

1116

1117

1118

1119

1120

1121

1122

1123

1124

1125

1126

1127

1128

1129

1130

1131

1132

1133

1134

1135

1136

1137

1138

1139

1140

1141

1142

1143

1144

1145

1146

1147

1148

1149

1150

1151

1152

1153

1154

1155

1156

1157

1158

1159

1160

1161

1162

1163

1164

1165

1166

1167

1168

1169

1170

1171

1172

1173

1174

1175

1176

1177

1178

1179

1180

1181

1182

1183

1184

1185

1186

1187

1188

1189

1190

1191

1192

1193

1194

1195

1196

1197

1198

1199

1200

1201

1202

1203

1204

1205

1206

1207

1208

1209

1210

1211

1212

1213

1214

1215

1216

1217

1218

1219

1220

1221

1222

1223

1224

1225

1226

1227

1228

1229

1230

1231

1232

1233

1234

1235

1236

1237

1238

1239

1240

1241

1242

1243

1244

1245

1246

1247

1248

1249

1250

1251

1252

1253

1254

1255

1256

1257

1258

1259

1260

1261

1262

1263

1264

1265

1266

1267

1268

1269

1270

1271

1272

1273

1274

1275

1276

1277

1278

1279

1280

1281

1282

1283

1284

1285

1286

1287

1288

1289

1290

1291

1292

1293

1294

1295

1296

1297

1298

1299

1300

1301

1302

1303

1304

1305

1306

1307

1308

1309

1310

1311

1312

1313

1314

1315

1316

1317

1318

1319

1320

1321

1322

1323

1324

1325

1326

1327

1328

1329

1330

1331

<p

The screenshot shows the IntelliJ IDEA interface with the BillingRestController.java file open. The code implements a REST controller for a bill repository. It includes methods for getting a bill by ID and a list of bills.

```
15  * @RestController
16  * public class BillingRestController {
17  *     private BillRepository billRepository;
18  *     private ProductItemRepository productItemRepository;
19  *     private CustomerRestClient customerRestClient;
20  *     private ProductItemRestClient productItemRestClient;
21  *
22  *     public BillingRestController(BillRepository billRepository, ProductItemRepository productItemRepository,
23  *                                 CustomerRestClient customerRestClient, ProductItemRestClient productItemRestClient) {
24  *         this.billRepository = billRepository;
25  *         this.productItemRepository = productItemRepository;
26  *         this.customerRestClient = customerRestClient;
27  *         this.productItemRestClient = productItemRestClient;
28  *
29  *     }
30  *
31  *     @GetMapping(path = "/fullBill/{id}")
32  *     public Bill getBill(@PathVariable(name = "id") Long id){
33  *         Bill bill=billRepository.findById(id).get();
34  *         Customer customer=customerRestClient.getCustomerById(bill.getCustomerID());
35  *         bill.setCustomer(customer);
36  *         bill.getProductItems().forEach(pi->{
37  *             Product product=productItemRestClient.getProductById(pi.getProductID());
38  *             // pi.setProduct(product); //get all product info
39  *             pi.setProductName(product.getName());
40  *         });
41  *         return bill;
42  *     }
43  *
44  * }
45  *
```

The screenshot shows the IntelliJ IDEA interface with the BillingServiceApplication.java file open. This is a Spring Boot application that starts up and performs database operations using repositories and rest clients.

```
22  * @SpringBootApplication
23  * @EnableFeignClients
24  * public class BillingServiceApplication {
25  *
26  *     public static void main(String[] args) {
27  *         SpringApplication.run(BillingServiceApplication.class, args);
28  *     }
29  *
30  *     @Bean
31  *     CommandLineRunner start(BillRepository billRepository,
32  *                            ProductItemRepository productItemRepository,
33  *                            CustomerRestClient customerRestClient,
34  *                            ProductItemRestClient productItemRestClient){
35  *         return args -> {
36  *             System.out.println("start");
37  *             Customer customer=customerRestClient.getCustomerById(1L);
38  *             Bill bill= billRepository.save(new Bill( id: null,new Date(), productItems: null, customer.getId()));
39  *             PagedModel<Product> productPagedModel= productItemRestClient.pageProducts();
40  *             productPagedModel.forEach(p ->{
41  *                 ProductItem productItem=new ProductItem();
42  *                 productItem.setPrice(p.getPrice());
43  *                 productItem.setQuantity(1+new Random().nextInt( bound: 100));
44  *                 productItem.setBill(bill);
45  *                 productItem.setProductID(p.getId());
46  *                 productItemRepository.save(productItem);
47  *             });
48  *         };
49  *     }
49  *
```

Eureka localhost:8888/BILLING-SERVICE +

localhost:8888/BILLING-SERVICE/fullBill/1

```
{
  "id": 1,
  "billingDate": "2020-12-26T10:07:32.339+00:00",
  "productItems": [
    {
      "id": 1,
      "quantity": 87,
      "price": 456,
      "productID": 1,
      "product": null,
      "productname": "ordin"
    },
    {
      "id": 2,
      "quantity": 87,
      "price": 786,
      "productID": 2,
      "product": null,
      "productname": "telef"
    },
    {
      "id": 3,
      "quantity": 19,
      "price": 234,
      "productID": 3,
      "product": null,
      "productname": "clavier"
    }
  ],
  "customerID": 1,
  "customer": {
    "id": 1,
    "name": "mohammed",
    "email": "mohammed@gmail.com"
  }
}
```

Authentication-service :

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help sec-service [C:\Users\user\IdeaProjects\sec-service] - AppRole.java

sec-service > src > main > java > org > sid > secserice > sec > entities > AppRole.java

Project 1 Project

sec-service C:\Users\user\IdeaProjects\sec-service:

- idea
- mvn
- src
 - main
 - java
 - org.sid.secserice
 - sec
 - entities
 - AppRole
 - AppUser
 - filters
 - JWTAuthenticationFilter
 - repo
 - AppRoleRepository
 - AppUserRepository
 - service
 - AccountService
 - AccountServiceimpl
 - web
 - AccountRestController.java
 - AccountRestController
 - RoleUserForm
 - SecurityConfig
 - SecServiceApplication
 - resources
 - static
 - templates
 - application.properties

AppRole.java

```

1 package org.sid.secserice.sec.entities;
2
3 import lombok.AllArgsConstructor;
4 import lombok.Data;
5 import lombok.NoArgsConstructor;
6 import lombok.ToString;
7
8 import javax.persistence.*;
9 import java.util.Collection;
10
11 @Entity
12 @Data @AllArgsConstructor @NoArgsConstructor @ToString
13 public class AppRole {
14     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
15     private Long id;
16     private String rolename;
17
18 }
```

Event Log

Lombok Requires Annotation Processing: Do you want to enable annotation processors? Enable (today 09:00)

The screenshot shows the IntelliJ IDEA interface with the project 'sec-service' open. The left sidebar displays the project structure, including 'src', 'main', 'java', 'org', 'sid', 'secservice', 'sec', 'entities', 'AppRole', 'AppUser', 'filters', 'repo', 'service', 'web', 'resources', 'static', 'templates', and 'application.properties'. The right pane shows the code for 'AppUser.java':

```
import com.fasterxml.jackson.annotation.JsonProperty;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.ToString;

import javax.persistence.*;
import java.util.ArrayList;
import java.util.Collection;

@Entity
@Data@AllArgsConstructor@NoArgsConstructor@ToString
public class AppUser {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @JsonProperty(access = JsonProperty.Access.READ_ONLY)
    private String username;
    private String password;
    @ManyToMany(fetch = FetchType.EAGER)
    private Collection<AppRole> appRoles=new ArrayList<>();
}
```

The screenshot shows the IntelliJ IDEA interface with the project 'sec-service' open. The left sidebar displays the project structure, including 'src', 'main', 'java', 'org', 'sid', 'secservice', 'sec', 'repo', 'AppRoleRepository', and 'findByRolename'. The right pane shows the code for 'AppRoleRepository.java':

```
package org.sid.secservice.sec.repo;

import org.sid.secservice.sec.entities.AppRole;
import org.springframework.data.jpa.repository.JpaRepository;

public interface AppRoleRepository extends JpaRepository<AppRole,Long> {
    AppRole findByRolename(String roleName);
}
```

The screenshot shows the IntelliJ IDEA interface with the project 'sec-service' open. The code editor displays the file `AppUserRepository.java`. The code defines an interface for managing users:

```
package org.sid.secservice.sec.repo;
import org.sid.secservice.sec.entities.AppUser;
import org.springframework.data.jpa.repository.JpaRepository;
public interface AppUserRepository extends JpaRepository<AppUser, Long> {
    AppUser findByUsername(String username);
}
```

The screenshot shows the IntelliJ IDEA interface with the project 'sec-service' open. The code editor displays the file `AccountService.java`. The code defines an interface for managing accounts:

```
package org.sid.secservice.sec.service;
import org.sid.secservice.sec.entities.AppRole;
import org.sid.secservice.sec.entities.AppUser;
import java.util.List;
public interface AccountService {
    AppUser addNewUser(AppUser appUser);
    AppRole addNewRole(AppRole appRole);
    void addRoleToUser(String username, String roleName);
    AppUser loadUserByUsername(String username);
    List<AppUser> listUsers();
}
```

IntelliJ IDEA interface showing the `AccountServiceImpl.java` file. The code implements the `AccountService` interface with methods for adding new users and roles.

```
1 package org.sid.secservice.sec.service;
2
3 import ...
4
5 @Service
6 @Transactional
7 public class AccountServiceImpl implements AccountService {
8     private AppUserRepository appUserRepository;
9     private AppRoleRepository appRoleRepository;
10
11     private PasswordEncoder passWordEncoder;
12
13     public AccountServiceImpl(AppUserRepository appUserRepository, AppRoleRepository appRoleRepository,
14         this.appUserRepository = appUserRepository;
15         this.appRoleRepository = appRoleRepository;
16         this.passWordEncoder = passWordEncoder;
17     }
18
19     @Override
20     public AppUser addNewUser(AppUser appUser) {
21         String pwd=appUser.getPassword();
22         appUser.setPassword(passWordEncoder.encode(pwd));
23         return appUserRepository.save(appUser);
24     }
25
26     @Override
27     public AppRole addNewRole(AppRole appRole) {
28
29
30
31
32
33
34
35
36 }
```

IntelliJ IDEA interface showing the `SecurityConfig.java` configuration class. It extends `WebSecurityConfigurerAdapter` and overrides the `configure` method to load users and set up security filters.

```
23
24
25
26 @Configuration
27 @EnableWebSecurity
28 public class SecurityConfig extends WebSecurityConfigurerAdapter {
29     @Autowired
30     private AccountService accountService;
31
32     @Override
33     protected void configure(AuthenticationManagerBuilder auth) throws Exception {
34         auth.userDetailsService(new UserDetailsService() {
35             @Override
36             public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
37                 AppUser appUser=accountService.loadUserByUsername(username);
38                 Collection<GrantedAuthority> authorities=new ArrayList<>();
39                 appUser.getAppRoles().forEach(e->{
40                     authorities.add(new SimpleGrantedAuthority(e.getRolename()));
41                 });
42                 return new User(appUser.getUsername(),appUser.getPassword(),authorities);
43             }
44         });
45
46     @Override
47     protected void configure(HttpSecurity http) throws Exception {
48         http.csrf().disable();
49         http.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS);
50         http.headers().frameOptions().disable();
51         http.authorizeRequests().antMatchers( ...antPatterns: "/h2-console/**").permitAll();
52         // http.formLogin();
53     }
54 }
```

The screenshot shows the IntelliJ IDEA interface with the project 'sec-service' open. The current file is 'SecurityConfig.java'. The code defines a configuration class for a Spring Security application. It includes methods for setting up CSRF protection, session creation policy, header options, and various security filters like 'JWTAuthenticationFilter'. It also defines an 'AuthenticationManager' bean.

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help sec-service [C:\Users\user\IdeaProjects\sec-service] - SecurityConfig.java  
sec-service > src > main > java > org > sid > sebservice > sec > SecurityConfig.java  
Project Z Structure Z Favorites Persistence Web  
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help sec-service [C:\Users\user\IdeaProjects\sec-service] - AccountRestController.java  
sec-service > src > main > java > org > sid > sebservice > sec > web > AccountRestController.java  
Project Z Structure Z Favorites Persistence Web  
1 package org.sid.secservice.sec.web;  
2  
3 import lombok.Data;  
4 import org.sid.secservice.sec.entities.AppRole;  
5 import org.sid.secservice.sec.entities.AppUser;  
6 import org.sid.secservice.sec.service.AccountService;  
7 import org.springframework.web.bind.annotation.*;  
8  
9 import java.util.List;  
10  
11 @RestController  
12 public class AccountRestController {  
13     private AccountService accountService;  
14  
15     public AccountRestController(AccountService accountService) { this.accountService = accountService; }  
16  
17     @GetMapping(path = "/users")  
18     public List<AppUser> appUsers() { return accountService.listUsers(); }  
19  
20     @PostMapping(path = "/users")  
21     public AppUser addUser(@RequestBody AppUser appUser) { return accountService.addNewUser(appUser); }  
22     @PostMapping(path = "/roles")  
23     public AppRole addRole(@RequestBody AppRole appRole) { return accountService.addNewRole(appRole); }  
24     @PostMapping(path = "/addRoleToUser")  
25     public void addRoleToUser(@RequestBody RoleUserForm roleUserForm){  
26         accountService.addRoleToUser(roleUserForm.getUsername(),roleUserForm.getRolename());  
27     }  
28 }  
29  
30  
31  
32  
33  
34  
35  
36
```

The screenshot shows the IntelliJ IDEA interface with the project 'sec-service' open. The current file is 'AccountRestController.java'. The code defines a REST controller for managing users and roles. It uses Lombok annotations for data classes and Spring's REST controller annotations. It includes methods for listing users, adding new users, adding new roles, and associating roles with users.

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help sec-service [C:\Users\user\IdeaProjects\sec-service] - AccountRestController.java  
sec-service > src > main > java > org > sid > sebservice > sec > web > AccountRestController.java  
Project Z Structure Z Favorites Persistence Web  
1 package org.sid.secservice.sec.web;  
2  
3 import lombok.Data;  
4 import org.sid.secservice.sec.entities.AppRole;  
5 import org.sid.secservice.sec.entities.AppUser;  
6 import org.sid.secservice.sec.service.AccountService;  
7 import org.springframework.web.bind.annotation.*;  
8  
9 import java.util.List;  
10  
11 @RestController  
12 public class AccountRestController {  
13     private AccountService accountService;  
14  
15     public AccountRestController(AccountService accountService) { this.accountService = accountService; }  
16  
17     @GetMapping(path = "/users")  
18     public List<AppUser> appUsers() { return accountService.listUsers(); }  
19  
20     @PostMapping(path = "/users")  
21     public AppUser addUser(@RequestBody AppUser appUser) { return accountService.addNewUser(appUser); }  
22     @PostMapping(path = "/roles")  
23     public AppRole addRole(@RequestBody AppRole appRole) { return accountService.addNewRole(appRole); }  
24     @PostMapping(path = "/addRoleToUser")  
25     public void addRoleToUser(@RequestBody RoleUserForm roleUserForm){  
26         accountService.addRoleToUser(roleUserForm.getUsername(),roleUserForm.getRolename());  
27     }  
28 }  
29  
30  
31  
32  
33  
34  
35  
36
```

IntelliJ IDEA screenshot showing the AccountRestController.java code. The code defines a controller for managing users and roles.

```
public AccountRestController(AccountService accountService) { this.accountService = accountService; }

@GetMapping(path = "/users")
public List<AppUser> appUsers() { return accountService.listUsers(); }

@PostMapping(path = "/users")
public AppUser addUser(@RequestBody AppUser appUser) { return accountService.addNewUser(appUser); }

@PostMapping(path = "/roles")
public AppRole addRole(@RequestBody AppRole appRole) { return accountService.addNewRole(appRole); }

@PostMapping(path = "/addRoleToUser")
public void addRoleToUser(@RequestBody RoleUserForm roleUserForm) {
    accountService.addRoleToUser(roleUserForm.getUsername(), roleUserForm.getRolename());
}

@Data
class RoleUserForm{
    private String username;
    private String rolename;
}
```

IntelliJ IDEA screenshot showing the JWTAuthenticationFilter.java code. This filter extends the UsernamePasswordAuthenticationFilter and handles user authentication.

```
public class JWTAuthenticationFilter extends UsernamePasswordAuthenticationFilter {

    private AuthenticationManager authenticationManager;

    public JWTAuthenticationFilter(AuthenticationManager authenticationManager) {
        this.authenticationManager = authenticationManager;
    }

    @Override
    public Authentication attemptAuthentication(HttpServletRequest request, HttpServletResponse response)
        System.out.println("attemptAuthentication");
        String username=request.getParameter( "username");
        String password=request.getParameter( "password");
        System.out.println(username);
        System.out.println(password);
        UsernamePasswordAuthenticationToken authenticationToken=
            new UsernamePasswordAuthenticationToken(username,password);
        return authenticationManager.authenticate(authenticationToken);
    }

    @Override
    protected void successfulAuthentication(HttpServletRequest request, HttpServletResponse response, FilterChain chain,
        System.out.println("successfulAuthentication");
        User user=(User) authResult.getPrincipal();
        Algorithm alg=Algorithm.HMAC256("mySecret123");
        String jwtAccessToken= JWT.create()
```

The screenshot shows the IntelliJ IDEA interface with the project 'sec-service' open. The code editor displays the `JWTAuthenticationFilter.java` file. The code implements a filter for handling successful authentication. It prints the username and password, creates a `UsernamePasswordAuthenticationToken`, and authenticates it using an `authenticationManager`. It then overrides the `successfulAuthentication` method to create a JWT token with a secret key ('mySecret123') and set it as a header in the response.

```
34     System.out.println(username);
35     System.out.println(password);
36     UsernamePasswordAuthenticationToken authenticationToken=
37         new UsernamePasswordAuthenticationToken(username,password);
38     return authenticationManager.authenticate(authenticationToken);
39 }
40 
41 @Override
42 protected void successfulAuthentication(HttpServletRequest request, HttpServletResponse response, FilterChain chain) throws IOException, ServletException {
43     System.out.println("successfulAuthentication");
44     User user=(User) authResult.getPrincipal();
45     Algorithm algo=Algorithm.HMAC256("mySecret123");
46     String jwtAccessToken= JWT.create()
47         .withSubject(user.getUsername())
48         .withExpiresAt(new Date(System.currentTimeMillis() + 5 * 60 * 1000))
49         .withIssuer(request.getRequestURI().toString())
50         .withClaim( "roles", user.getAuthorities().stream().map(ga->ga.getAuthority()).collect(Collectors.toList()));
51     .sign(algo);
52     response.setHeader("Authorization", jwtAccessToken);
53 }
54 }
```

The screenshot shows the IntelliJ IDEA interface with the project 'sec-service' open. The code editor displays the `SecServiceApplication.java` file. It contains a `CommandLineRunner` named `start` that takes an `AccountService` dependency. Inside the `start` method, roles are added to the service: `USER`, `ADMIN`, `CUSTOMER_MANAGER`, `PRODUCT_MANAGER`, and `BILLS_MANAGER`. Then, users are added with their respective roles and passwords. Finally, roles are assigned to each user.

```
22     @Bean
23     PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }
24 
25     @Bean
26     CommandLineRunner start(AccountService accountService) {
27         return args -> {
28             accountService.addNewRole(new AppRole(null, rolename: "USER"));
29             accountService.addNewRole(new AppRole(null, rolename: "ADMIN"));
30             accountService.addNewRole(new AppRole(null, rolename: "CUSTOMER_MANAGER"));
31             accountService.addNewRole(new AppRole(null, rolename: "PRODUCT_MANAGER"));
32             accountService.addNewRole(new AppRole(null, rolename: "BILLS_MANAGER"));
33 
34             accountService.addNewUser(new AppUser(null, username: "admin", password: "123", new ArrayList<>());
35             accountService.addNewUser(new AppUser(null, username: "user1", password: "123", new ArrayList<>());
36             accountService.addNewUser(new AppUser(null, username: "user2", password: "123", new ArrayList<>());
37             accountService.addNewUser(new AppUser(null, username: "user3", password: "123", new ArrayList<>());
38             accountService.addNewUser(new AppUser(null, username: "user4", password: "123", new ArrayList<>());
39 
40             accountService.addRoleToUser(username: "user1", rolename: "USER");
41             accountService.addRoleToUser(username: "admin", rolename: "USER");
42             accountService.addRoleToUser(username: "admin", rolename: "ADMIN");
43             accountService.addRoleToUser(username: "user2", rolename: "USER");
44             accountService.addRoleToUser(username: "user2", rolename: "CUSTOMER_MANAGER");
45             accountService.addRoleToUser(username: "user3", rolename: "USER");
46             accountService.addRoleToUser(username: "user3", rolename: "PRODUCT_MANAGER");
47             accountService.addRoleToUser(username: "user4", rolename: "USER");
48             accountService.addRoleToUser(username: "user4", rolename: "BILLS_MANAGER");
49         };
50     };
51 }
```

ARC

HTTP request

Socket

History

POST http://localhost:8080/login

Yesterday

POST http://localhost:8080/login

GET http://localhost:8080

mardi 17 novembre 2020

PUT http://localhost:8082/produit/4

GET http://localhost:8082/produit/4

POST http://localhost:8082/produit

GET http://localhost:8082/produit

GET http://localhost:8082/produit/1

Saved

Save a request and recall it from here

Install new ARC with new features!

Headers Body Variables

Body content type application/x-www-form-urlencoded Editor view Form data (www-url-form-encoded)

ENCODE PAYLOAD DECODE PAYLOAD

username admin

password 123

ADD FORM PARAMETER

200 OK 435.84 ms DETAILS ^

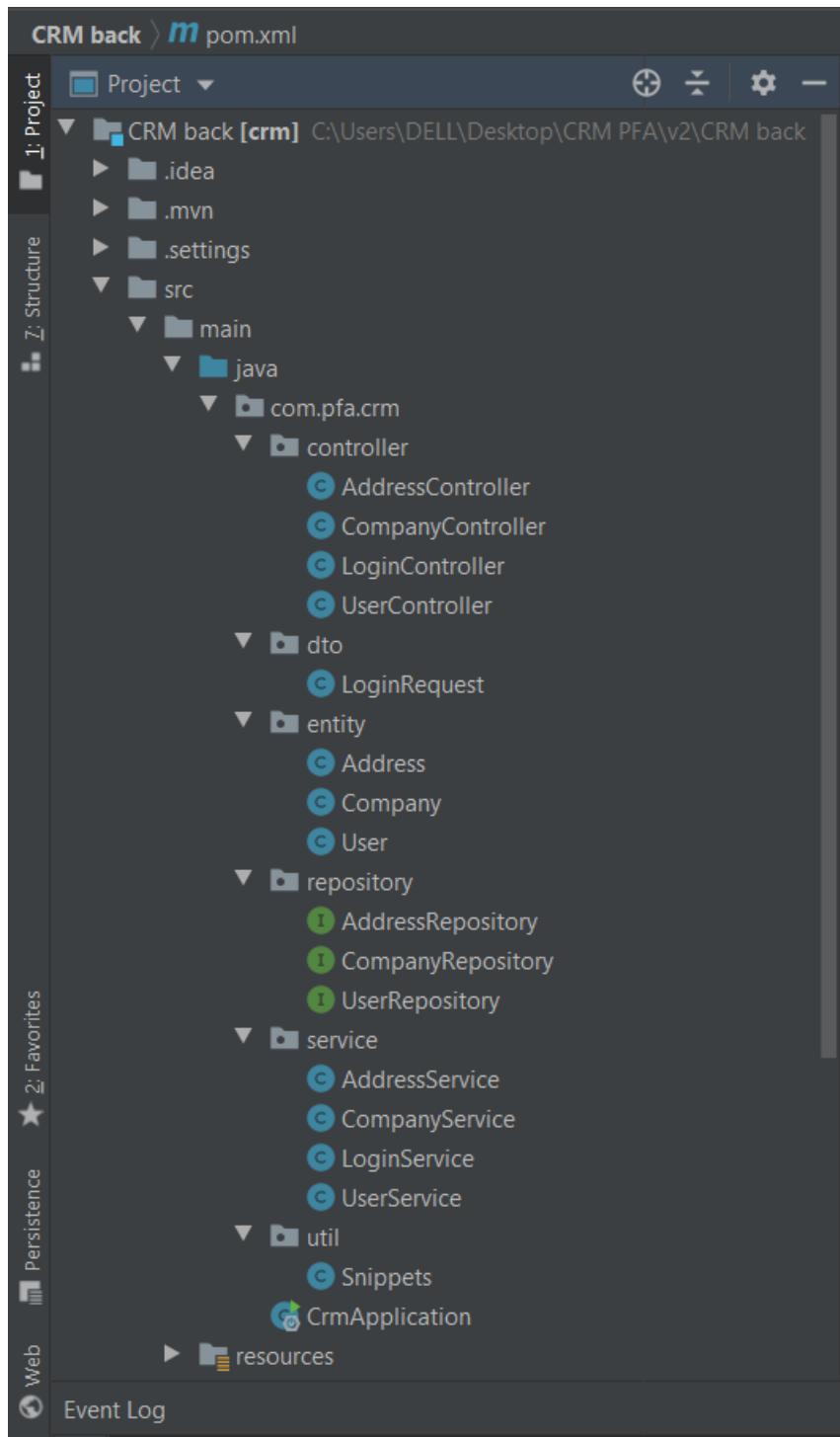
POST http://localhost:8080/login

Response headers (8) Request headers (1) Redirects (0) Timings

```
author: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZG1pbjIsInJvbGViIjpbIkFETU10IiwiVVNFUiJdLCJpc3MiO
zation: iIvbG9naW4iLCJleHAiOjQ4MjY5Mzc4NTY2MzUwMH0.8e6F_XTwhtgJOFmOLEiESe2n_gCfo91CZPG3X7DJB3E
Content-Type: application/json; charset=UTF-8
Content-Length: 0
Date: Sat, 26 Dec 2020 10:41:25 GMT
Server: Apache/2.4.41 (Ubuntu)
Set-Cookie: PHPSESSID=1234567890; expires=Sat, 26-Dec-2020 10:41:25 UTC; path=/; secure; HttpOnly
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Content-Length: 0
Date: Sat, 26 Dec 2020 10:41:25 GMT

```

Selected environment: Default ⓘ



```
▼ {
  "_links": {
    "suppliers": {
      "href": "http://localhost:8083/suppliers{?page,size,sort}",
      "templated": true
    },
    "profile": {
      "href": "http://localhost:8083/profile"
    }
  }
}
```

Raw Parsed

Partie Front – end

