TP - SPRING CRM API

CONSIGNES :

Dans ce TP votre objectif sera de créer une API en respectant une **architecture REST** vue en classe cette semaine. Cette API au travers de différentes URL vous permettra de réaliser un **CRUD complet** autour de plusieurs entités :

- **Customer**
- **Order**
- **User**

CONFIGURATION DU PROJET ET STARTERS - pom.xml

- spring-boot-starter-data-jpa
- spring-boot-starter-web
- postgresql
- spring-boot-devtools
- mapstruct

CREATION DU PROJET - spring initializr

Cette application ne disposant pas d'interface web, vous pourrez créer un projet maven générant un fichier .jar en sortie.

CREATION DES ENTITÉS - package model

Vous devrez alors créer ces trois entités en définissant à l'intérieur les propriétés suivantes :

- **POUR L'ENTITÉ CUSTOMER**

id **(Integer)**
lastName **(String)**
firstName **(String)**
company **(String)**
mail **(String)**
phone **(String)**
mobile **(String)**
notes **(String)**
active **(Boolean)**
orders **(Set<Order>)**

- **POUR L'ENTITÉ ORDER**

Id **(Integer)**
adrEt **(Double)**
numberOfDays **(Double)**
tva **(Double)**
status **(String)**
type **(String)**
notes **(String)**
customer **(Customer)**

- **POUR L'ENTITÉ USER**

Id **(Integer)**
password **(String)**
mail **(String)**
grants **(String)**

Pour chaque entité vous devrez :
- implémenter un **constructeur vide sans argument**
- Implémenter un **constructeur permettant d'initialiser tous les champs** de la classe
- La fonction **toString()**
- Implémenter les mutateurs et accesseurs

Pour chaque entité vous utiliserez les annotations suivantes :
- @Entity
- @Table
- @Id
- @GeneratedValue
- @Column

CREATION DES SERVICES - package service

Vous devrez alors créer pour chaque entité un service permettant une interaction avec la couche repository, en charge de la persistance des données en BDD.

INTERFACES -

CustomerService
getAll()
getId()
createCustomer()
deleteCustomer()
updateCustomer()

OrderService
getAll()
getId()
createOrder()
deleteOrder()
updateOrder()

UserService
getAll()

getId()
createUser()
deleteUser()
updateUser()

IMPLEMENTATIONS - package service.impl

CustomerServiceImpl
OrderServiceImpl
UserServiceImpl

CREATION DES REPOSITORIES - package repository

Chaque repository devra implémenter l'interface JpaRepository pour bénéficier des méthodes CRUD classiques à savoir :

- findAll()
- findById()
- save()
- delete()

CustomerRepository

OrderRepository

UserRepository

CREATION DE L'API- package api.v1

Dans ces classes, vous utiliserez les annotations suivantes :

- @RestController
- @RequestMapping
- @CrossOrigin
- @Autowired
- @GetMapping
- @ApiOperation
- @PostMapping
- @DeleteMapping
- @PutMapping
- @PatchMapping

Vous utiliserez la classe ResponseEntity pour retourner les réponses de chaques méthodes.

CustomerApi
OrderApi
UserApi

Exemple CustomerApi:

Dans cette classes, lorsque vous allez créer un Customer en BDD, vous devrez mapper votre CustomerDto en Customer.

Lorsque vous récupérerez un Customer en BDD vous devrez mapper un Customer en CustomerDto.

CREATION DES DTO - package api.dto

Dans cette partie vous devrez créer les objets de transfert de données aussi appelés DTO pour simplifier les transferts de données entre les sous-systèmes d'une application logicielle.

CustomerDto

OrderDto

UserDto

CREATION DES CLASSES MAPPER - package mapper

Dans ces classes vous utiliserez les annotations suivantes pour gérer au mieux le mapping des entités :

- @Component
- @Mapper
- @Mapping

CustomerMapper

```
CustomerDto mapCustomerToCustomerDto(Customer customer);
Customer mapCustomerDtoToCustomer(CustomerDto customerDto);
```

OrderMapper

```
OrderDto mapOrderToOrderDto(Order order);
Order mapOrderDtoToOrder(OrderDto orderDto);
```

UserMapper

```
UserDto mapUserToUserDto(User user);
User mapUserDtoToUser(UserDto userDto);
```

CREATION DE LA BASE DE DONNÉES - package resources.db

Vous retrouverez avec les sources de ce TP le fichier create_tables.sql dans lequel vous retrouverez la base de données et ses tables ainsi que plusieurs données que nous chargerons pour cette application.

Vous devrez pour cela :

- Créer une base de données PostreSQL via PGADMIN4
- DBeaver pour connecter et administrer cette base de données

Vous n'êtes pas obligé de lancer la création de la BDD au lancement de l'application. Vous pouvez le faire à part sur les logiciels mentionnés ci-dessus.

CREATION DE LA CLASSE D'EXCEPTION - package resources.db

Créer une classe UnknownResourceException qui se chargera de renvoyer les exceptions.

CONFIGURATION DU FICHIER PROPERTIES - package resources.application.properties

Vous devrez configurer le fichier application.properties en utilisant les propriétés suivantes :

spring.datasource.url=
spring.datasource.name=
spring.datasource.username=
spring.datasource.password=
spring.datasource.driver-class-name=

spring.jpa.hibernate.ddl-auto=
spring.jpa.properties.hibernate.dialect=
spring.jpa.defer-datasource-initialization=

CONFIGURATION DES LOGS - package resources.application.properties

Dans le fichier application.properties :

logging.file.path=/opt/data/crm/logs
logging.file.name=crm.log

INFORMATIONS PROJET - package resources.application.properties

Dans le fichier application.properties :

info.project.name=@project.name@
info.project.desc=@project.description@
info.project.version=@project.version@

DOCUMENTEZ VOS PROJETS AVEC SWAGGER 2 - package config

Swagger est capable de générer une documentation détaillée au format JSON, répondant aux spécifications OpenAPI.

Pour plus d'informations :
https://openclassrooms.com/fr/courses/4668056-construisez-des-microservices/7652911-documentez-votre-microservice-avec-swagger-2

EXEMPLES D'OBJET JSON RENVOYÉS PAR VOS URLS

http://localhost:8080/v1/users

[{"id":1,"username":"admin","mail":"admin@test.fr","grants":"ADMIN"}]

http://localhost:8080/v1/customers

[{"id":1,"lastname":"JONES","firstname":"Indiana","company":"Université de Chicago","mail":"indiana.jonas@univ-chicago.com","mobile":"0666666666","notes":"Les notes d'Indiana","active":true,"orders":[{"id":1,"label":"Formation Java","adrEt":450.0,"numberOfDays":5.0,"tva":20.0,"status":"En cours","type":"Forfait","notes":"Test","customerId":1},{"id":2,"label":"Formation Spring","adrEt":450.0,"numberOfDays":3.0,"tva":20.0,"status":"En attente","type":"Forfait","notes":"Test","customerId":1}]},
{"id":2,"lastname":"KENOBI","firstname":"Obi-Wan","company":"Jedis","mail":"obiwan.kenobi@jedis.com","mobile":"0666666666","notes":"Les notes d'Obi Wan","active":true,"orders":[{"id":3,"label":"Formation Jedi","adrEt":1500.0,"numberOfDays":2.0,"tva":20.0,"status":"Payée","type":"Forfait","notes":"Notes sur la formation","customerId":2}]},
{"id":3,"lastname":"MCCLANE","firstname":"John","company":"NYPD","mail":"john.mcclane@nypd.com","mobile":"0666666666","notes":"Les notes de John","active":false,"orders":[]},
{"id":4,"lastname":"MCFLY","firstname":"Marty","company":"DOC","mail":"marty.mcfly@doc.com","notes":"Les notes de Marty","active":false,"orders":[]}]

http://localhost:8080/v1/orders

[{"id":1,"label":"Formation Java","adrEt":450.0,"numberOfDays":5.0,"tva":20.0,"status":"En cours","type":"Forfait","notes":"Test","customerId":1},{"id":3,"label":"Formation Jedi","adrEt":1500.0,"numberOfDays":2.0,"tva":20.0,"status":"Payée","type":"Forfait","notes":"Notes sur la formation","customerId":2},{"id":2,"label":"Formation Spring","adrEt":450.0,"numberOfDays":3.0,"tva":20.0,"status":"En attente","type":"Forfait","notes":"Test","customerId":1}]

TESTER VOTRE API - Advanced Rest Client

http://localhost:8080/v1/customers

http://localhost:8080/v1/customers/1 (GET)
http://localhost:8080/v1/customers (POST)
http://localhost:8080/v1/customers/1 (DELETE)
http://localhost:8080/v1/customers/1 (PUT)
http://localhost:8080/v1/customers/1 (PATCH)

http://localhost:8080/v1/orders

http://localhost:8080/v1/orders/1 (GET)
http://localhost:8080/v1/orders (POST)
http://localhost:8080/v1/orders/1 (DELETE)
http://localhost:8080/v1/orders/1 (PUT)
http://localhost:8080/v1/orders/1 (PATCH)

http://localhost:8080/v1/users

http://localhost:8080/v1/users/1 (GET)
http://localhost:8080/v1/users (POST)
http://localhost:8080/v1/users/1 (DELETE)
http://localhost:8080/v1/users/1 (PUT)
http://localhost:8080/v1/users/1 (PATCH)