2. Conception REST

Indiquer la requête ou la séquence de requêtes nécessaires pour réaliser les opérations suivantes :

• Lister les annonces

GET /annonces

• Lister les annonces du 54

GET /annonces?dep=54

• Lister les annonces de la catégorie 'voitures' 4

GET /categories?nom=voiture GET /annonces?idcateg=id

• Créer une catégorie

POST /categories

```
json : {"id":"ZHAKHldnladn","nom":"nouv","descr":"nouveau"}
```

• Modifier une annonce existante

PUT /annonces/{idannonce}

```
json : {"text":"je suis un nouveau text"}
```

• Créer une annonce, l'associer à une catégorie et un département

POST /annonces

```
json : {"titre":"nouveau", "date":"nouvelledate", "text":"nouveau"..., "idcateg":"idcateg", "iddep":"iddep"}
```

Modifier la catégorie d'une annonce

PUT /annonces/{idannonce}

```
json : {"idcateg":"idcateg"}
```

• Ajouter une image à une annonce

PUT /annonces/{idannonce}

```
json : {"idimage":"idimage"}
```

• Supprimer une annonce et toutes les images associées

DELETE /photos/{idannonce}

3. Conception du projet

1 - Schéma UML des ressources utilisées par le service de prise de commandes

Pour composer les items et les orders

Clients

id : int
name : String
email : String
password : String
total_spent : double
fidelity_points : int
created_at : datetime
updated_at : datetime

Categories

id : int label : String description : String

> 1..* | Categories | 0..*

Sandwiches

id : int name : String description : String bread : String price : double bio : boolean

Orders id: int name: String mail: String client id: int amount_to_pay : double discount : double token: String status: int payment ref: String payement_method : int payment_date : datetime withdraw: datetime created_at : datetime updated_at : datetime items 1..* Items id: int uri : String label: String price: double quantity: int order id:int

2 Identifier les URI des différentes ressources, et les actions possibles sur chacune d'elles en leur associant une méthode HTTP.

```
/orders - GET POST
/orders/{idOrders} - PATCH
/sandwiches - GET POST
/sandwiches/{idSandwiches) - PATCH DELETE
/categories - GET POST
/categories/{idCategories} - PATCH DELETE
/users - GET POST
```

/users/{idUsers} - PATCH DELETE

3. Illustrer ces actions en déroulant un scénario dans lequel un client crée une commande, modifie sa date de livraison, effectue le paiement puis interroge le service pour connaître l'état de cette commande ; à chaque étape, expliciter la requête et la réponse Rest (méthode, uri, status, données, headers)

body json {"status": 2}

```
Créer une commande :
POST /orders
body json {"step":"à payer","withdraw":"2023-03-01 19:20", "amount_to_pay":"7.50",
"user":"4256", "sandwiches": {4,8}}
body json {
"name": "Sacquard",
"mail": "Oubbi@gmail.com",
"client_id": 141,
"amount_to_pay" : 3.30,
"discount": NULL,
"token": "2983d045d6026814ba0fe811b6cc535b0d3a65bfebb761645c38b51d5bd392".
"status": 1,
"payment_ref": NULL,
"payment date": NULL,
"withdraw": "2023-03-01 19:20"
réponse : 201 created
Modifier sa date de livraison :
PATCH /orders/{idOrders}
body json {"withdraw":"2023-03-03 13:01"}
réponse : 200 OK
Effectuer le paiement :
PATCH /orders/{idOrders}
```

réponse : 200 OK

Connaitre état commande :

GET /orders/{idOrders}/status

réponse : 200 OK