Gestione Comanda

Il microservizio Gestione Comanda si occupa principalmente di gestire gli ordini dei clienti (microservizio GestioneCliente) e fornire alla cucina (microservizio GestioneCucina) gli ordini da preparare.

La comunicazione con gli altri microservizi avviene tramite Message Broker come segue:

- Il microservizio GestioneCliente comunica verso GestioneComanda tramite il topic Kafka NotifyOrderEvent.
- Il microservizio GestioneComanda comunica verso GestioneCucina tramite il topic Kafka SendOrderEvent.
- Il microservizio GestioneCucina comunica verso GestioneComanda tramite il topic Kafka NotifyPrepEvent.

Il microservizio GestioneComanda è sprovvisto di un componente HTTP Controller nella sua Interfaccia (che contiene solo EventController), viene quindi creato un controller di TEST per interagire direttamente con i componenti del servizio ai soli fini di test.

Le API di test riguardano principalmente il Message Broker e il DataBase.

Message Broker

E' possibile interagire direttamente con i tre topic (NotifyOrderEvent, SendOrderEvent, NotifyPrepEvent) tramite operazioni di GET e di POST:

- GET: le chiamate GET all'indirizzo `.../test/{topic}` restituiscono l'ultimo messaggio passato sul topic specificato;
- POST: le chiamate POST all'indirizzo `.../test/{topic}` permettono di iniettare dall'esterno dei messaggi sul topic specificato, necessitano di un corpo in formato JSON che equivale alla serializzazione dell'oggetto che si aspetta quel topic.

POST Post to topic sendOrderEvent

```
localhost:8080/test/sendorderevent
```

API di POST con la quale è possibile iniettare all'interno del broker oggetti al fine di test.

Si testa il topic sendOrderEvent da gestione comanda verso gestione cucina.

Parametri della richiesta (body JSON):

- `id` (numero intero, opzionale) : Identificatore dell'ordine (autogenerato in ogni caso dal sistema)
- `idComanda` (numero intero, obbligatorio) : Identificatore della comanda di cui l'ordine fa parte
- `idPiatto` (stringa, obbligatorio) : Identificatore del piatto ordinato dal cliente
- `stato` (numero intero tra 0 e 3, obbligatorio) : Stato dell'ordine (0: Ordine preso in carico, 1: Ordine in coda di preparazione, 2: Ordine in preparazione, 3: Ordine preparato)
- `tOrdinazione` (stringa timestamp di pattern "yyyy-MM-dd HH:mm:ss.SSS", opzionale): Istante temporale in cui viene effettuata l'ordinazione
- `urgenzaCliente` (numero intero tra 0 e 2, obbligatorio) : Attributo urgenza del cliente (0 : espresso non urgenza, 1 : espresso urgenza, 2 : default)

Risposta:

- `id` : Identificatore dell'ordine
- `idComanda` : Identificatore della comanda di cui l'ordine fa parte
- `idPiatto` : Identificatore del piatto ordinato dal cliente
- `stato` : Stato dell'ordine (0: Ordine preso in carico, 1: Ordine in coda di preparazione, 2: Ordine in preparazione, 3: Ordine preparato)
- `tOrdinazione` : Istante temporale in cui viene effettuata l'ordinazione
- `urgenzaCliente` : Attributo urgenza del cliente (0 : espresso non urgenza, 1 : espresso urgenza, 2 : default)

Body raw (json)

json

```
{
    "idComanda":7,
    "idPiatto":"SUH724",
    "stato":1,
    "urgenzaCliente":0
}
```

GET Get from topic sendOrderEvent

localhost:8080/test/sendorderevent

API di GET con la quale è possibile ottenere l'ultimo messaggio letto sul topic SendOrderEvent.

Si testa il topic notifyPrepEvent da gestione cucina verso gestione comanda

Risposta:

json


```
{
    "id": 7,
    "idComanda": 7,
    "idPiatto": "SUH724",
    "stato": 1,
    "urgenzaCliente": 0,
    "tordinazione": "2024-04-30 22:03:17.199"
}
```

POST Post to topic notifyOrderEvent

localhost:8080/test/notifyorderevent

API di POST con la quale è possibile iniettare all'interno del broker oggetti al fine di test.

Si testa il topic notifyOrderEvent da gestione cliente verso gestione comanda.

Parametri della richiesta (body JSON):

- `id` (numero intero, obbligatorio) : Identificatore dell'ordine
- `idComanda` (numero intero, obbligatorio) : Identificatore della comanda di cui l'ordine fa parte

Risposta:

- `id` : Identificatore dell'ordine
- `idComanda` : Identificatore della comanda di cui l'ordine fa parte

Body raw (json)

json

```
{
    "id": 1,

    "idComanda": 4
}
```

GET Get from topic notifyOrderEvent

```
localhost:8080/test/notifyorderevent
```

Espone una API di GET con la quale è possibile ottenere l'ultimo messaggio letto sul topic NotifyOrderEvent.

Si testa il topic notifyPrepEvent da gestione cucina verso gestione comanda.

Risposta:

- `id` : Identificatore dell'ordine
- `idComanda` : Identificatore della comanda di cui l'ordine fa parte

---

POST Post to topic notifyPrepEvent

```
localhost:8080/test/notifyprepevent
```

API di POST con la quale è possibile iniettare all'interno del broker oggetti al fine di test.

Si testa il topic notifyPrepEvent da gestione cucina verso gestione comanda.

Parametri della richiesta (body JSON):

- `id` (numero intero, obbligatorio) : Identificatore dell'ordine
- `idComanda` (numero intero, obbligatorio) : Identificatore della comanda di cui l'ordine fa parte
- `stato` (numero intero tra 0 e 3, obbligatorio) : Stato dell'ordine (0: Ordine preso in carico, 1: Ordine in coda di preparazione, 2: Ordine in preparazione, 3: Ordine preparato)

Risposta:

- `id` : Identificatore dell'ordine
- `idComanda` : Identificatore della comanda di cui l'ordine fa parte
- `stato` : Stato dell'ordine (0: Ordine preso in carico, 1: Ordine in coda di preparazione, 2: Ordine in

preparazione, 3: Ordine preparato)

Body raw (json)

```
json
```

```
{
    "id": 1,
    "idComanda": 4,
    "stato": 2
}
```

GET Get from topic notifypPepEvent

localhost:8080/test/notifyprepevent

API di GET con la quale è possibile ottenere l'ultimo messaggio letto sul topic NotifyPrepEvent.

Si testa il topic notifyPrepEvent da gestione cucina verso gestione comanda.

Risposta:

- `id` : Identificatore dell'ordine
- `idComanda` : Identificatore della comanda di cui l'ordine fa parte
- `stato` : Stato dell'ordine (0: Ordine preso in carico, 1: Ordine in coda di preparazione, 2: Ordine in preparazione, 3: Ordine preparato)

Database

Le API di test verso il databse riguardano unicamente l'entità Ordine, vengono testati:

- l'inserimento nel database;
- la ricerca di un elemento;
- la modifica parziale di un elemento;
- la ricerca di tutti gli ordini per una data comanda;
- la rimozione di un ordine

POST Post order

localhost:8080/test/order

Salva nel database l'oggetto ordine dato un ordineDTO

Parametri della richiesta (body JSON):

- `id` (numero intero, opzionale) : Identificatore dell'ordine (autogenerato in ogni caso dal sistema)
- `idComanda` (numero intero, obbligatorio) : Identificatore della comanda di cui l'ordine fa parte
- `idPiatto` (stringa, obbligatorio) : Identificatore del piatto ordinato dal cliente
- `stato` (numero intero tra 0 e 3, obbligatorio) : Stato dell'ordine (0: Ordine preso in carico, 1: Ordine in coda di preparazione, 2: Ordine in preparazione, 3: Ordine preparato)
- `tOrdinazione` (stringa timestamp di pattern "yyyy-MM-dd HH:mm:ss.SSS", opzionale): Istante temporale in cui viene effettuata l'ordinazione
- `urgenzaCliente` (numero intero tra 0 e 2, obbligatorio) : Attributo urgenza del cliente (0 : espresso non urgenza, 1 : espresso urgenza, 2 : default)

Risposta:

- `id` : Identificatore dell'ordine
- `idComanda` : Identificatore della comanda di cui l'ordine fa parte
- `idPiatto` : Identificatore del piatto ordinato dal cliente
- `stato` : Stato dell'ordine (0: Ordine preso in carico, 1: Ordine in coda di preparazione, 2: Ordine in preparazione, 3: Ordine preparato)
- `tOrdinazione` : Istante temporale in cui viene effettuata l'ordinazione
- `urgenzaCliente` : Attributo urgenza del cliente (0 : espresso non urgenza, 1 : espresso urgenza, 2 : default)

Body raw (json)

json


```
{
    "idComanda":4,
    "idPiatto":"PIA770",
    "stato":0,
    "urgenzaCliente":1
```

```
}
```

GET Get order by id

```
localhost:8080/test/order/1
```

Restituisci l'ordine corrispondente all'id dato in input

Parametri della query string:

- `id` (obbligatorio): Identificatore dell'ordine da recuperare

Risposta:

- `id` : Identificatore dell'ordine
- `idComanda` : Identificatore della comanda di cui l'ordine fa parte
- `idPiatto` : Identificatore del piatto ordinato dal cliente
- `stato` : Stato dell'ordine (0: Ordine preso in carico, 1: Ordine in coda di preparazione, 2: Ordine in preparazione, 3: Ordine preparato)
- `tOrdinazione` : Istante temporale in cui viene effettuata l'ordinazione
- `urgenzaCliente` : Attributo urgenza del cliente (0 : espresso non urgenza, 1 : espresso urgenza, 2 : default)

PATCH Patch order

```
localhost:8080/test/order/1
```

Aggiornamento parziale dell'entità ordine, è possibile fornire solamente gli oggetti da aggiornare.

Parametri della query string:

- `id` (obbligatorio): Identificatore dell'ordine da recuperare

Parametri della richiesta (body JSON):

- `stato` (numero intero tra 0 e 3, opzionale) : Stato dell'ordine (0: Ordine preso in carico, 1: Ordine in coda di preparazione, 2: Ordine in preparazione, 3: Ordine preparato)
- `urgenzaCliente` (numero intero tra 0 e 2, opzionale) : Attributo urgenza del cliente (0 : espresso non urgenza, 1 : espresso urgenza, 2 : default)

Risposta:

- `id` : Identificatore dell'ordine
- `idComanda` : Identificatore della comanda di cui l'ordine fa parte
- `idPiatto` : Identificatore del piatto ordinato dal cliente
- `stato` : Stato dell'ordine (0: Ordine preso in carico, 1: Ordine in coda di preparazione, 2: Ordine in preparazione, 3: Ordine preparato)
- `tOrdinazione` : Istante temporale in cui viene effettuata l'ordinazione
- `urgenzaCliente` : Attributo urgenza del cliente (0 : espresso non urgenza, 1 : espresso urgenza, 2 : default)

Body raw (json)

json

```
{
    "stato": 1,
    "urgenzaCliente": 1
}
```

GET Get orders by IdComanda

localhost:8080/test/orders/4

Restituisce una lista con tutti gli ordini relativi a una data comanda

Parametri della query string:

- `idComanda` (obbligatorio): Identificatore della comanda di cui gli ordini fanno parte

Risposta:

Lista di oggetti con i seguenti parametri:

- `id` : Identificatore dell'ordine
- `idComanda` : Identificatore della comanda di cui l'ordine fa parte
- `idPiatto` : Identificatore del piatto ordinato dal cliente
- `stato` : Stato dell'ordine (0: Ordine preso in carico, 1: Ordine in coda di preparazione, 2: Ordine in preparazione, 3: Ordine preparato)
- `tOrdinazione` : Istante temporale in cui viene effettuata l'ordinazione
- `urgenzaCliente` : Attributo urgenza del cliente (0 : espresso non urgenza, 1 : espresso urgenza, 2 : default)

DELETE Delete order

```
localhost:8080/test/order/8
```

Cancella l'ordine con il dato ID dal databse

Parametri della query string:

- `id` (obbligatorio): Identificatore dell'ordine da recuperare

Risposta:

```
Plain Text
```

```
204 No Content
```