EVENTS WORKFLOW

Documentation générale sur les événements échangés entre les micro-services

Auteurs : Gaetan Duminy, Rudy Meersman, Nassim Bounouas, Nikita Rousseau
Dernière mise à jour : 11/11/2018

Remarques générales

Cette documentation suit une pseudo-notation :

```
#micro-service

SYNCHRONICITY:
HTTP_METHOD on PATH API Gateway -> (ACTION : TOPIC - PAYLOAD) ->
SERVICE -> (ACTION_RESPONSE  : TOPIC - id request + payload)
```

Cette règle de définition des workflows permet de nous assurer qu’il existe au moins un workflow qui permet de réaliser chaque fonctionnalité.

L’architecture est construite de sorte à être asynchrone. L’ensemble des appels pour interroger le système passe au travers d’une API Gateway. Cette API Gateway permet de produire des messages sur le Bus et initier un workflow. Chaque appel à l’API Gateway est identifié de manière unique par un GUID. Cette identification permet d’associer une réponse à une requête.

En effet, lors d’un appel à l’API Gateway, on ne répond pas de manière immédiate à la requête : on génère dynamiquement une URL de callback où le client pourra périodiquement aller chercher la réponse à sa requête. Il pourra effectuer une boucle d’attente active en fonction du code de retour http.

La documentation permettant d’interroger l’API Gateway est disponible à la racine de celle-ci, ou encore dans le fichier `swagger.yaml` à la racine du dépôt.

#restaurant_service

Asynchrone : POST on /restaurants API Gateway -> (RESTAU) -> Restaurant Service -> (RESTAURANT_LIST_RESPONSE : restaurant - id request - List<Restaurant> (id restaurant + name + address))

Response on : GET on /restaurants?id=X

#menu_service

Asynchrone : Post /restaurant-menu API Gateway -> (FOOD_MENU_REQUEST : restaurant - id request + id restaurant) -> Menu Service -> (FOOD_MENU_RESPONSE : restaurant - id request - List<Menu> (id meal + name))

Response on : GET on /restaurant-menu?id=X

#ordering_service

Asynchrone: Post /order API Gateway -> (RESTAURANT_ORDER_REQUEST : ordering - id restaurant - id meal - client name - client address - id code(?)) -> Ordering (save the order)

Response on : GET on /order?id=X

#ordering_service

Asynchrone: Post /validate-order?id=X API Gateway -> (VALIDATE_ORDER_REQUEST: ordering - id order) -> Ordering -> (ORDER_STATUS : id request + status {pending - in progress - ready for pickup})

Response on : GET on /order-status?id=X

As Gail or Erin, I can order my lunch from a restaurant so that the food is delivered to my place

#menu_service

Asynchrone : POST on /list_categories API Gateway ->(CATEGORY_LIST_REQUEST : restaurant - id request) -> Menu Service -> (CATEGORY_LIST_RESPONSE : restaurant - id request - List<Categorie>(name))

Response on : GET on /list_categories?id=X

#menu_service

Asynchrone : POST on /list_meals_by_category API Gateway -> (FOOD_LIST_REQUEST : restaurant - id categorie + id request) -> Menu Service -> (FOOD_LIST_RESPONSE : restaurant - id request + MAP<Menu>(id restaurant, List<Food>(id + name)))

Response on : GET on /list_meals_by_category?id=X

As Gail, I can browse the food catalogue by categories so that I can immediately identify my favorite junk food

#eta

Asynchrone : POST on /eta API Gateway -> (ETA_REQUEST :eta - id request + from + to) -> Eta Service -> (ETA_RESPONSE : eta - id request + Eta(int))

Response on : GET on /eta?id=X

As Erin, I want to know before ordering the estimated time of delivery of the meal so that I can schedule my work around it, and be ready when it arrives

#paywall

Async: POST on /paywall API Gateway -> (PAYMENT_PLACED : order(ordering ?) - id order + card number) -> PAYWALL SERVICE-> (PAYMENT_ACCEPTED : order - id request) -> (VALIDATE_ORDER_REQUEST: ordering - id order) -> Ordering

Response on : GET on /paywall?id=X

As Erin, I can pay directly by credit card on the platform, so that I only have to retrieve my food when delivered

#ordering

Asynchrone : POST on /order_list_by_restaurant API Gateway -> (ORDER_LIST_REQUEST : restaurant - id request + id restaurant) -> Ordering -> (ORDER_LIST_RESPONSE : restaurant - id request + List <Order>(id food + name))

Response on : GET on /order_list_by_restaurant?id=X

As Jordan, I want to access to the order list, so that I can prepare the meal efficiently

#delivery

Async: POST on /map API Gateway -> (MAP_DELIVERY_PROBE : delivery - long + lat) -> DELIVERY SERVICE-> (MAP_DELIVERY_AVAILABLE_PICKUPS : delivery - id request + gps + List<orders>)

Response on : GET on /map?id=X

As Jamie, I want to know the orders that will have to be delivered around me, so that I can choose one and go to the restaurant to begin the course

#delivery

Asynchrone : POST on /notify_delivery API Gateway -> (NOTIFY_DELIVERY_REQUEST : delivery - id request + id order) -> Delivery Service

-> (NOTIFY_DELIVERY_RESPONSE : restaurant - id restaurant - id order) -> Ordering Service

-> (NOTIFY_DELIVERY_RESPONSE : payment - id steed - id order - amount) -> Payment Service

As Jamie, I want to notify that the order has been delivered, so that my account can be credited and the restaurant can be informed

#restaurant

Asynchrone : POST on/meal_feedback API Gateway -> (FEEDBACK_REQUEST : restaurant - id meal + id restaurant + comment) -> Restaurant Service

As Jordan, I want the customers to be able to review the meals so that I can improve them according to their feedback

#delivery

Asynchrone : POST on /delivery_location API Gateway -> (DELIVERY_LOCATION_PUSH : delivery - id order + id steed + long + lat) -> Delivery Service

GET on /delivery_location API Gateway -> (DELIVERY_LOCATION_REQUESTED : delivery - id order) -> Delivery Service -> (DELIVERY_LOCATION_STATUS : delivery - id order - lastLatitude - lastLongitude - timestamp)

Note:
think path callable from mobile app with geo coordinates and id coursier + id order

As a customer (Gail, Erin), I want to track the geolocation of the coursier in real time, so that I can anticipate when I will eat

#delivery

Async: GET on /stats API Gateway -> (STEED_STAT_REQUEST : delivery - id_steed) -> DELIVERY SERVICE -> (DELIVERY_STAT_RESPONSE : delivery - MAP<coursier, field> => VALUE)

As Terry, I want to get some statistics (speed, cost) about global delivery time and delivery per coursier

Asynchrone : POST on /create_promotional_code API Gateway -> (CREATE_CODE_REQUEST : restaurant - id request +code + reduction) -> Restaurant

As Terry, I can emit a promotional code so that I can attract more customer to my restaurant.

Asynchrone : POST on /steed_status API Gateway -> (SEND_STEED_STATUS : delivery - id_steed - status) -> Delivery Service

As Jamie, I want to inform quickly that I can't terminate the course (accident, sick), so that the order can be replaced.

Asynchrone : POST on /add_promotional_code API Gateway -> (ADD_PROMOTIONAL_CODE : restaurant - id_request + code +List< id food> + reduction) -> Menu Service

As Terry, I can emit a promotional code based on my menu contents (e.g., 10% discout for an entry-main course-dessert order), so that I can sell more expensive orders

Asynchrone : GET on /delivery_location API Gateway -> (DELIVERY_LOCATION_REQUESTED : delivery - id order) -> Delivery Service -> (ETA_UPDATE_REQUESTED : delivery- id order - lastLatitude - lastLongitude - timestamp) ->Eta service -> (DELIVERY_LOCATION_STATUS : delivery - id request + id order + Eta(int) + to + lastLongitude + timestamp)

As Gail or Erin, I can follow the position of Jamie in real time, so that the food ETA can be updated automatically