microservices (calculation services):

we will recive order object. From “pub/sub”

recivce from sub: order entitie: id, restaurantid, totalPrice, orderDatetime , DeliveringDatetime, customerId, deliveringadresse,

Table: entity agent bonus: agentid, amount, DeliveringDatetime, order id

Table entity mtogo earning: ? amount, datetime, order id

Table entity restaurant earning ?: amount, datetime, orderid

we have 3 main logic:

1. Mtogo earning: metogo fee: based on order.totalrpice

First 100 Kr 16%

Amount to 500 Kr 15%

Amount to 1.000 Kr 14%

Anything more 13%

1. restaurant earning:

order.total price – mtogo earning.

1. Agent bonus:

Employees will get 5 % as bonus from Mtogo.amount value before VAT (moms) only between 18:00-06:00

microservices (Restauant services):

* + **Responsibilities**:
    - **Enter adress then shows all restaurants in that area (rest api)**
    - Displaying all restaurant information. (rest api)
    - Then choose a restaurant and then display menus (rest api)
    - Receiving and confirming orders from MTOGO.
    - Handling food preparation status updates (receive from payment sub).
    - Handling food preparation status updates (done) (pub)
  + **Entities**: Restaurant, Order Details(list of all selected items from the menu).
  + **Value object**: menu items (in this case we will avoid menu but connect menu items with a restaurant)
  + **Interactions**: Works with Order Management Context for receiving orders and Delivery Management Context for notifying when an order is ready.

microservices (Order services):

* + **Responsibilities**:
    - create a order (order id, customer id, restaurant id , ordered time, total price, from order details, status if delivered or not)
    - create a order details(list of menu items that choosed from the specific restaurant with their price, and which order it belongs to)
    - Processing mock payment confirmation.
    - pub orderdetails
  + **Entities**: Order, Orderdetails, mock payment gateway simple api endpoint which always accept the payment. Pub new order created and payment went through
  + **Interactions**: Interfaces with the Restaurant Management Context.

microservices (Delivering services):

* + **Responsibilities**:
    - create a agent (agent id, name)
    - create a orderDelivering (order id, )
    - create
    - create a order details(list of menu items that choosed from the specific restaurant with their price, and which order it belongs to)
    - Processing mock payment confirmation.
    - pub orderdetails
  + **Entities**: Order, Orderdetails, Leveringdetails, mock payment gateway simple api endpoint which always accept the payment. Pub new order created and payment went through
  + **Interactions**: Interfaces with the Restaurant Management Context.

Kode:

1. Kode – logic – entity – value object – aggregate – error handling ? (\*)
2. Test
3. Design pattern (Filter – Port - Adapter) (\*)
4. Pub-sub (Hvad forventer du at modtag ? sub) (hvad forventer du at sende afsted ? pub) (\*)
5. Rest api (\*)
6. Dockerfile (image) (\*)
7. Database connection (hvordan er bedst at gøre med docker compose) ? (\*)
8. Api gateway (kræver at opsætte først google cloud – vent indtil vider) (\*)

Opsætning - github:

1. Projekt – user stories osv.
2. Git-action – til main
   1. CI dvs. Test alt
   2. CD derefter deploy det.

Opsætning – google cloud:

1. Install Cgloud – sæt vores user/project osv. (\*)
2. Opret login(rolebased) (\*)
3. Api gateway (helt til sidst, når vi har sat Kubernetes op, pubsub og alle endpoints osv.)
4. Deploy vores project – dvs. flere server, kubernetes, pubsub (\*)

Lige nu:

Mtogo repo:

1. deploy – docker compose – (hvordan sætter det op i google ? smider vi en tom mappe med yml eller kun yml file som docker compose ? hvad med kubnetes ?)

Calculation

1. pub/sub (hvordan skriver vi kode ? hvad sætter vi det sammen i google med topic ?)
2. CI/CD (Github test først inden og når det ind på main deploy det i google server)
3. docker – (vi har lavet file, men hvordan bliver den image lavet af sig selv efter vi har deployet det i google ?)

Restaurant

1. pub/sub
2. CI/CD
3. docker
4. deploy – docker compose

lille frontend – login

test apigateway

vi ved hvordan pub/sup virker

den virker -> CI/CD (vores test går igennem)

docker og docker compose med kubneters er setup.

FROM mcr.microsoft.com/dotnet/sdk:8.0@sha256:35792ea4ad1db051981f62b313f1be3b46b1f45cadbaa3c288cd0d3056eefb83 AS build-env

WORKDIR /App

# Copy everything

COPY . ./

# Restore as distinct layers

RUN dotnet restore

# Build and publish a release

RUN dotnet publish -c Release -o out

# Build runtime image

FROM mcr.microsoft.com/dotnet/aspnet:8.0@sha256:6c4df091e4e531bb93bdbfe7e7f0998e7ced344f54426b7e874116a3dc3233ff

WORKDIR /App

COPY --from=build-env /App/out .

ENTRYPOINT ["dotnet", "CalculationManagementService.dll"]

docker build -t

services:

user-service:

build:

context: ./UserService

ports:

- "5000:5000"

order-service:

build:

context: ./OrderService

ports:

- "5001:5000"

* + 1. KODE
    2. Pubsub / rest endpoint
    3. Kubernetes
    4. CICD
    5. Api Gateway
    6. En liile frontend

APIGatway setup:   
  
swagger: '2.0'

info:

title: Multi-Microservice API

description: API Gateway configuration for Restaurant, Order, Delivery, Calculation, and Analytics services with REST and Pub/Sub

version: 1.0.0

schemes:

- https

produces:

- application/json

paths:

# RestaurantManagementMicroservice

/restaurant/menu/{itemId}:

get:

summary: Get menu item from Restaurant Management

operationId: getMenuItem

parameters:

- in: path

name: itemId

type: string

required: true

description: ID of the menu item

responses:

'200':

description: Menu item retrieved successfully

x-google-backend:

address: https://restaurant-management.namespace.svc.cluster.local

path\_translation: APPEND\_PATH\_TO\_ADDRESS

/restaurant/pubsub/menu-updates:

post:

summary: Publish menu update to Pub/Sub

operationId: publishMenuUpdate

parameters:

- in: body

name: message

schema:

type: object

properties:

topic:

type: string

description: Pub/Sub topic name

menuItemId:

type: string

description: Updated menu item ID

responses:

'200':

description: Menu update published successfully

x-google-backend:

address: https://pubsub.googleapis.com/v1/projects/YOUR\_PROJECT\_ID/topics/{topic}:publish

protocol: https

# OrderManagementMicroservice

/order/create:

post:

summary: Create a new order

operationId: createOrder

parameters:

- in: body

name: order

required: true

schema:

type: object

properties:

customerId:

type: string

items:

type: array

items:

type: string

responses:

'201':

description: Order created successfully

x-google-backend:

address: https://order-management.namespace.svc.cluster.local

path\_translation: APPEND\_PATH\_TO\_ADDRESS

/order/pubsub/order-status:

post:

summary: Publish order status update to Pub/Sub

operationId: publishOrderStatus

parameters:

- in: body

name: message

schema:

type: object

properties:

topic:

type: string

description: Pub/Sub topic name

orderId:

type: string

description: Order ID

status:

type: string

description: Updated status

responses:

'200':

description: Order status published successfully

x-google-backend:

address: https://pubsub.googleapis.com/v1/projects/YOUR\_PROJECT\_ID/topics/{topic}:publish

protocol: https

# DeliveryManagementService

/delivery/assign:

post:

summary: Assign delivery to an agent

operationId: assignDelivery

parameters:

- in: body

name: delivery

required: true

schema:

type: object

properties:

orderId:

type: string

deliveryAgentId:

type: string

responses:

'200':

description: Delivery assigned successfully

x-google-backend:

address: https://delivery-management.namespace.svc.cluster.local

path\_translation: APPEND\_PATH\_TO\_ADDRESS

/delivery/pubsub/delivery-tracking:

post:

summary: Publish delivery tracking update to Pub/Sub

operationId: publishDeliveryTracking

parameters:

- in: body

name: message

schema:

type: object

properties:

topic:

type: string

description: Pub/Sub topic name

trackingId:

type: string

description: Tracking ID

location:

type: string

description: Current location

responses:

'200':

description: Delivery tracking published successfully

x-google-backend:

address: https://pubsub.googleapis.com/v1/projects/YOUR\_PROJECT\_ID/topics/{topic}:publish

protocol: https

# CalculationManagementMicroservice

/calculation/calculate-bonus:

post:

summary: Calculate delivery bonus

operationId: calculateBonus

parameters:

- in: body

name: bonusDetails

required: true

schema:

type: object

properties:

deliveryAgentId:

type: string

completedOrders:

type: integer

responses:

'200':

description: Bonus calculated successfully

x-google-backend:

address: https://calculation-management.namespace.svc.cluster.local

path\_translation: APPEND\_PATH\_TO\_ADDRESS

/calculation/pubsub/calculate-report:

post:

summary: Publish calculation report to Pub/Sub

operationId: publishCalculationReport

parameters:

- in: body

name: message

schema:

type: object

properties:

topic:

type: string

description: Pub/Sub topic name

reportId:

type: string

description: Report ID

responses:

'200':

description: Calculation report published successfully

x-google-backend:

address: https://pubsub.googleapis.com/v1/projects/YOUR\_PROJECT\_ID/topics/{topic}:publish

protocol: https

# AnalyticsReportingService

/analytics/report:

get:

summary: Get analytics report

operationId: getAnalyticsReport

responses:

'200':

description: Analytics report retrieved successfully

x-google-backend:

address: https://analytics-reporting.namespace.svc.cluster.local

path\_translation: APPEND\_PATH\_TO\_ADDRESS

/analytics/pubsub/event-log:

post:

summary: Publish analytics event log to Pub/Sub

operationId: publishAnalyticsEventLog

parameters:

- in: body

name: message

schema:

type: object

properties:

topic:

type: string

description: Pub/Sub topic name

eventId:

type: string

description: Event ID

eventData:

type: string

description: Event details

responses:

'200':

description: Event log published successfully

x-google-backend:

address: https://pubsub.googleapis.com/v1/projects/YOUR\_PROJECT\_ID/topics/{topic}:publish

protocol: https