Rapport sur le mobile SDK et Cloud API DJI

NB : Quel que soit le SDK à utiliser, il faut accéder au site officiel de DJI pour se connecter en tant que développeur, puis créer sa propre application selon le SDK utilisé afin d'obtenir son APP KEY et son APP ID.

I- Phase de développement de la Backend :

code du fichier .env :

```
DJI_ACCESS_TOKEN="f43d5f1ec9ca103e2bdea4aeace9509"
DJI_MISSION_URL="https://cloud-api.dji.com/api/wayline/v1/missions"
DJI_TELEMETRY_URL="https://cloud-api.dji.com/v1/telemetry"
DJI_STREAM_URL="https://cloud-api.dji.com/v1/live/start"
```

(les urls de ce fichier sont des urls non fonctionnelles)

code fichier config.py :

```
from dotenv import load_dotenv
import os
#load_dotenv(override=True)
load_dotenv()
class Settings:
    DJI_ACCESS_TOKEN: str = os.getenv('DJI_ACCESS_TOKEN')
    DJI_MISSION_URL: str = os.getenv('DJI_MISSION_URL')
    DJI_TELEMETRY_URL: str = os.getenv('DJI_TELEMETRY_URL')
    DJI_STREAM_URL: str = os.getenv('DJI_STREAM_URL')
settings = Settings()
print(f"DJI_ACCESS_TOKEN: {settings.DJI_ACCESS_TOKEN}")
print(f"DJI_MISSION_URL: {settings.DJI_MISSION_URL}")
print(f"DJI_TELEMETRY_URL: {settings.DJI_TELEMETRY_URL}")
print(f"DJI_STREAM_URL: {settings.DJI_STREAM_URL}")
```

code fichier models.py :

```
from pydantic import BaseModel
from typing import List

class Waypoint(BaseModel):
        latitude: float
        longitude: float
        altitude:float= 100
        speed:float = 5

class MissionRequest(BaseModel):
        mission_name: str
        waypoints: List[Waypoint]
```

code fichier services.py :

```
import requests
import logging
from config import settings

HEADERS = {
   "Authorization": f"Bearer {DJI_ACCESS_TOKEN}",
   "Content-Type": "application/json"}

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

def send_mission(mission_data):
   try:
      logging.info(f"Envoi de la mission à {DJI_API_URL}")
      response = requests.post(DJI_API_URL, json=mission_data, headers=HEADERS,
timeout=10)
      response.raise_for_status()
   except requests.exceptions.ConnectionError as e:
      logging.error(f"Erreur de connexion : {e}")
      return {"error": "Impossible de contacter l'API DJI"}
   except requests.exceptions.Timeout:
      logging.error("Le serveur DJI ne répond pas (timeout)")
      return {"error": "Le serveur DJI ne répond pas"}
   except requests.exceptions.RequestException as e:
      logging.error(f"Erreur HTTP : {e}")
      return {"error": f"Erreur API DJI : {str(e)}"}

def get_telemetry():
   try:
      response = requests.get(settings.DJI_TELEMETRY_URL, headers=HEADERS)
      response.raise_for_status()
      return response.json()
   except requests.exceptions.RequestException as e:
      logger.error(f"Erreur lors de la récupération de la télémétrie : {e}")
      if response is not None:
         logger.error(f"Réponse de l'API : {response.status_code} - {response.text}")
      return {"error": f"Erreur lors de la récupération de la télémétrie : {str(e)}"}

def start_stream():
   try:
      response = requests.post(settings.DJI_STREAM_URL, headers=HEADERS)
      response.raise_for_status()
      return response.json()
   except requests.exceptions.RequestException as e:
      logger.error(f"Erreur lors du démarrage du streaming : {e}")
      if response is not None:
         logger.error(f"Réponse de l'API : {response.status_code} - {response.text}")
      return {"error": f"Erreur lors du démarrage du streaming : {str(e)}"}
```

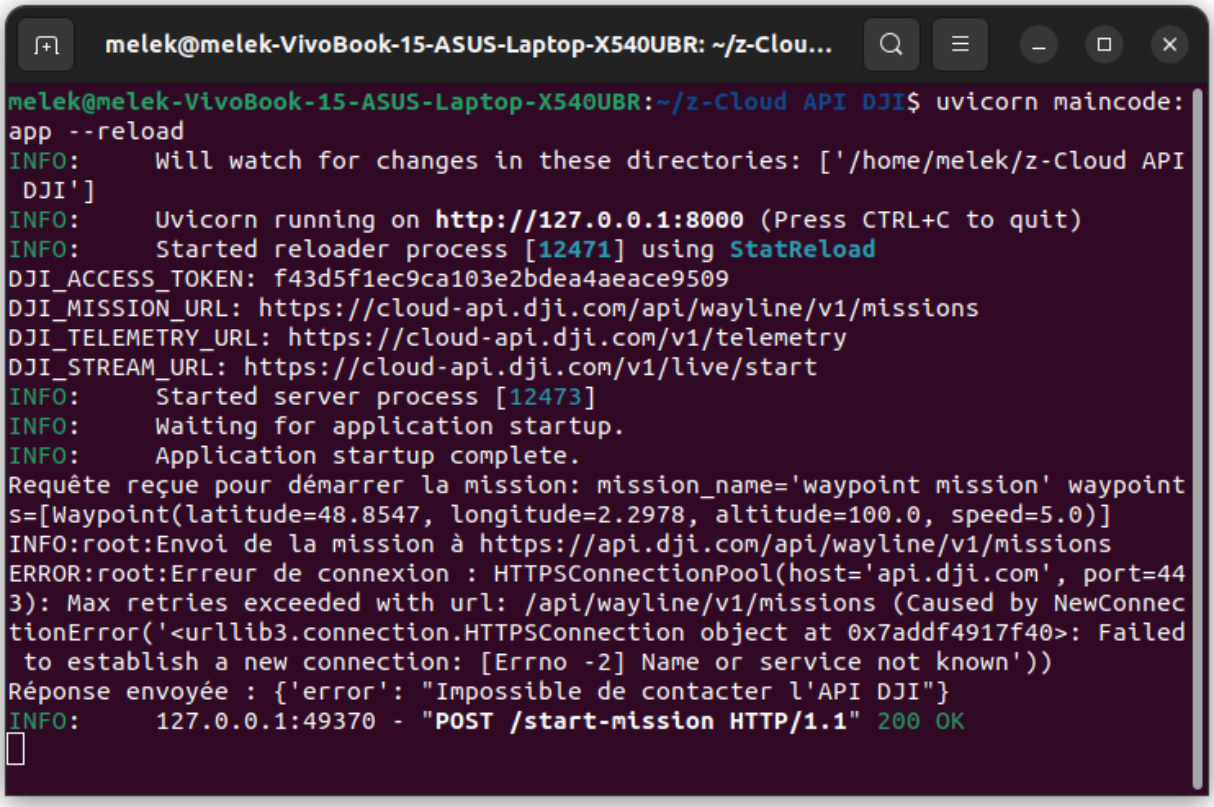
code fichier maincode,py :

```
from fastapi import FastAPI
from models import MissionRequest
from config import settings
from services import send_mission, get_telemetry, start_stream
from fastapi.middleware.cors import CORSMiddleware

app = FastAPI()

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

@app.post("/start-mission")
async def start_mission(mission: MissionRequest):
    print("Requête reçue pour démarrer la mission:", mission)
    response = send_mission(mission.dict())
    print("Réponse envoyée :", response)
    return response

@app.get("/get-telemetry")
async def get_drone_telemetry():
    """
    Cette route récupère la télémétrie du drone.
    """
    return get_telemetry()

@app.get("/start-stream")
async def start_drone_stream():
    """
    Cette route démarre le streaming du drone.
    """
    return start_stream()

@app.get("/")
def read_root():
    return {"message": "Bienvenue sur l'API DJI"}
```

---

puis il faut exécuter la commande : uvicorn maincode:app –reload

Nous avons reçu le message suivant, donc notre backend est fonctionnel et nous avons un problème au niveau des URLs de la DJI Cloud API.



II- les étapes pour intégrer le cloud :

les étapes suivantes sont sur le cite officiel de dji pour le Cloud API.

Partie installation :

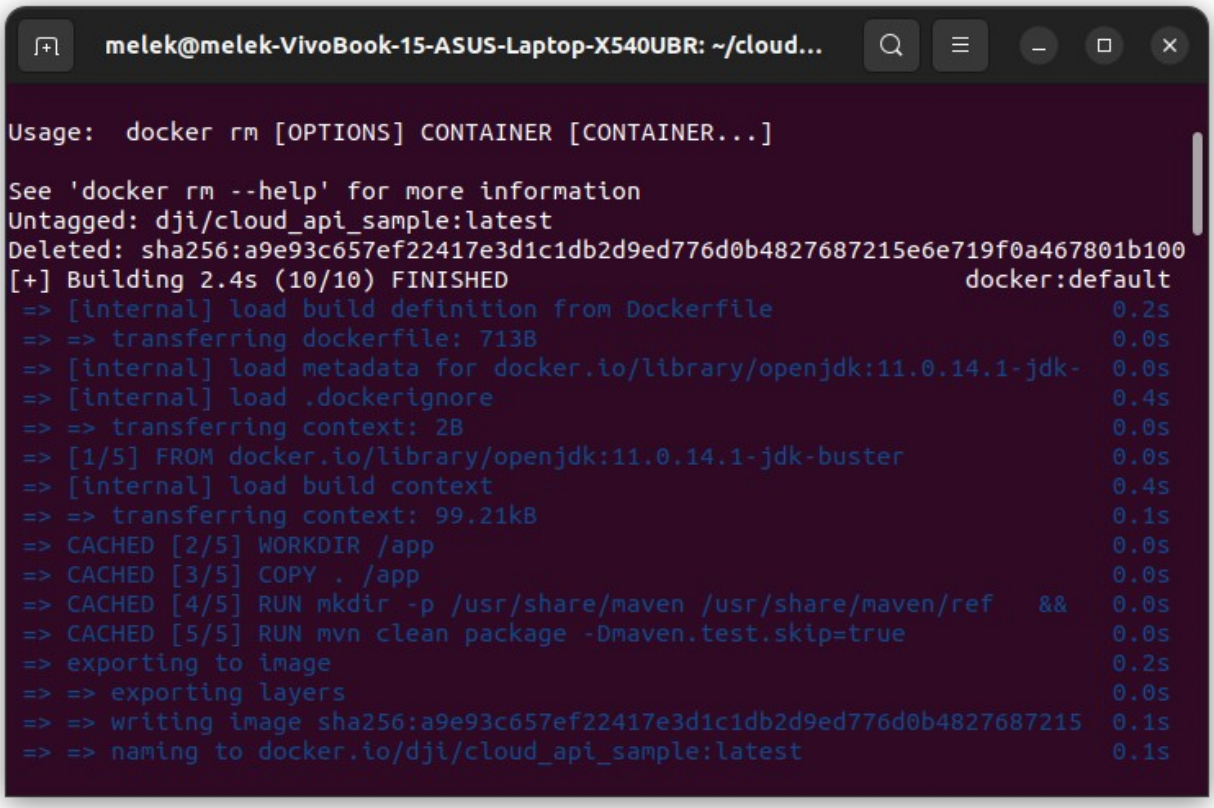
- java version 11 ou supérieur
- MySQL version 8.0.26
- EMQX version 4.4.0
- Redis version 6.2
- Nginx version 1.20.2
- installer docker
- installer docker compose
- télécharger le docker source code depuis github

les commandes pour tout les étapes de cette partie sont disponibles sur :
https://developer.dji.com/doc/cloud-api-tutorial/en/quick-start/source-code-deployment-steps.html
https://developer.dji.com/doc/cloud-api-tutorial/en/quick-start/docker-deployment-steps.html

pour les étapes ce cette partie nous avons :(sans erreur)

- bien installer notre base de donnée avec mysql
- bien charger le fichier images avec docker

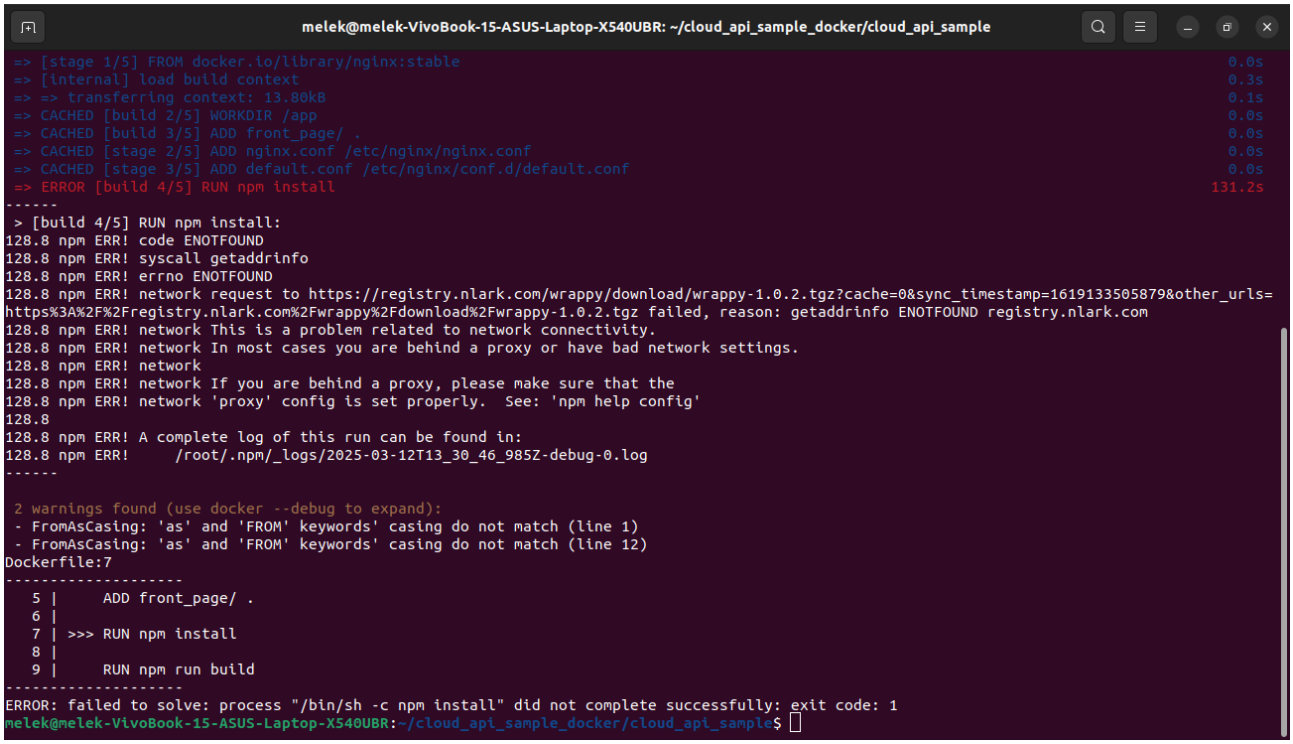
pour la configuration du fichier config.ts (source/nginx/front_page/src/api/http) nous avons faire le changement des variables puis exécuter la commande ./update_backend.sh (remarque : le url utilisé pour ce fichier le celle de notre backend de la partie 1 https://127,0,0,1:8000/)le résultats est la suivante :



Pour la configuration du fichier application.yml (source/backend_service/sample/src/main/resources) nous avons fait les modifications suivantes :

- url: jdbc:mysql://localhost:3306/cloud_sample?useSSL=false&allowPublicKeyRetrieval=true
  pour le url du datasource
- host: localhost -------- > pour le redis host
- host: 192.168.1.1 puis host : 127.0.0.1 -------- > pour le mqtt

la résultat des modifications est la suivante :



Comme nous allons rencontrer des erreurs à l’étape précédente, nous ne pouvons pas appliquer la commande sudo docker-compose up -d pour démarrer le conteneur.