

# 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.

```
melek@melek-VivoBook-15-ASUS-Laptop-X540UBR: ~/z-Clou...
melek@melek-VivoBook-15-ASUS-Laptop-X540UBR:~/z-Cloud API DJI$ uvicorn maincode:
app --reload
INFO: Will watch for changes in these directories: ['/home/melek/z-Cloud API
DJI']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [12471] using StatReload
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
INFO: Started server process [12473]
INFO: Waiting for application startup.
INFO: Application startup complete.
Requête reçue pour démarrer la mission: mission_name='waypoint mission' waypoint
s=[Waypoint(latitude=48.8547, longitude=2.2978, altitude=100.0, speed=5.0)]
INFO:root:Envoi de la mission à https://api.dji.com/api/wayline/v1/missions
ERROR:root:Erreur de connexion : HTTPSConnectionPool(host='api.dji.com', port=44
3): Max retries exceeded with url: /api/wayline/v1/missions (Caused by NewConnec
tionError('<urllib3.connection.HTTPSConnection object at 0x7addf4917f40>: Failed
to establish a new connection: [Errno -2] Name or service not known'))
Réponse envoyée : {'error': "Impossible de contacter l'API DJI"}
INFO: 127.0.0.1:49370 - "POST /start-mission HTTP/1.1" 200 OK
```

## II- les étapes pour intégrer le cloud :

les étapes suivantes sont sur le site 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 toutes 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 de cette partie nous avons :(sans erreur )

- bien installer notre base de données avec mysql
- bien charger les fichiers images avec docker

pour la configuration du fichier config.ts (source/nginx/front\_page/src/api/http) nous avons fait le changement des variables puis exécuter la commande ./update\_backend.sh (remarque : le url utilisé pour ce fichier est la même que notre backend de la partie 1 https://127,0,0,1:8000/)le résultat est le suivant :

```
melek@melek-VivoBook-15-ASUS-Laptop-X540UBR: ~/cloud...
Usage:  docker rm [OPTIONS] CONTAINER [CONTAINER...]

See 'docker rm --help' for more information
Untagged: dji/cloud_api_sample:latest
Deleted: sha256:a9e93c657ef22417e3d1c1db2d9ed776d0b4827687215e6e719f0a467801b100
[+] Building 2.4s (10/10) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.2s
=> => transferring dockerfile: 713B                                0.0s
=> [internal] load metadata for docker.io/library/openjdk:11.0.14.1-jdk- 0.0s
=> [internal] load .dockerignore                                  0.4s
=> => transferring context: 2B                                      0.0s
=> [1/5] FROM docker.io/library/openjdk:11.0.14.1-jdk-buster      0.0s
=> [internal] load build context                                  0.4s
=> => transferring context: 99.21kB                                0.1s
=> CACHED [2/5] WORKDIR /app                                       0.0s
=> CACHED [3/5] COPY . /app                                        0.0s
=> CACHED [4/5] RUN mkdir -p /usr/share/maven /usr/share/maven/ref  && 0.0s
=> CACHED [5/5] RUN mvn clean package -Dmaven.test.skip=true      0.0s
=> exporting to image                                              0.2s
=> => exporting layers                                             0.0s
=> => writing image sha256:a9e93c657ef22417e3d1c1db2d9ed776d0b4827687215 0.1s
=> => naming to docker.io/dji/cloud_api_sample:latest             0.1s
```

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 :

```
melek@melek-VivoBook-15-ASUS-Laptop-X540UBR: ~/cloud_api_sample_docker/cloud_api_sample
=> [stage 1/5] FROM docker.io/library/nginx:stable                0.0s
=> [internal] load build context                                  0.3s
=> => transferring context: 13.80kB                                0.1s
=> CACHED [build 2/5] WORKDIR /app                                0.0s
=> CACHED [build 3/5] ADD front_page/ .                          0.0s
=> CACHED [stage 2/5] ADD nginx.conf /etc/nginx/nginx.conf       0.0s
=> CACHED [stage 3/5] ADD default.conf /etc/nginx/conf.d/default.conf 0.0s
=> ERROR [build 4/5] RUN npm install                             131.2s
-----
> [build 4/5] RUN npm install:
128.8 npm ERR! code ENOTFOUND
128.8 npm ERR! syscall getaddrinfo
128.8 npm ERR! errno ENOTFOUND
128.8 npm ERR! network request to https://registry.nlark.com/wrappy/download/wrappy-1.0.2.tgz?cache=0&sync_timestamp=1619133505879&other_urls=https%3A%2F%2Fregistry.nlark.com%2Fwrappy%2Fdownload%2Fwrappy-1.0.2.tgz failed, reason: getaddrinfo ENOTFOUND registry.nlark.com
128.8 npm ERR! network This is a problem related to network connectivity.
128.8 npm ERR! network In most cases you are behind a proxy or have bad network settings.
128.8 npm ERR! network
128.8 npm ERR! network If you are behind a proxy, please make sure that the
128.8 npm ERR! network 'proxy' config is set properly. See: 'npm help config'
128.8
128.8 npm ERR! A complete log of this run can be found in:
128.8 npm ERR! /root/.npm/_logs/2025-03-12T13_30_46_985Z-debug-0.log
-----
2 warnings found (use docker --debug to expand):
- FromAsCasing: 'as' and 'FROM' keywords' casing do not match (line 1)
- FromAsCasing: 'as' and 'FROM' keywords' casing do not match (line 12)
Dockerfile:7
-----
5 |     ADD front_page/ .
6 |
7 | >>> RUN npm install
8 |
9 |     RUN npm run build
-----
ERROR: failed to solve: process "/bin/sh -c npm install" did not complete successfully: exit code: 1
melek@melek-VivoBook-15-ASUS-Laptop-X540UBR: ~/cloud_api_sample_docker/cloud_api_sample$
```

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.