# OSHapp — Guide de passation et réutilisation du code

Version: 2025-10-11

## 1) Liens de livraison à compléter

* **Lien GitHub (version fédérée, code source complet)**: (https://github.com/Abdellatif444/OSHapp-Application)

## 2) Architecture et composants

* **Frontend**: Flutter (Web servi via Nginx)
  + Dossier: frontend/
  + Dockerfile: frontend/Dockerfile (build Flutter → Nginx nginx:alpine)
  + Port: 3000
* **Backend**: Spring Boot (Java 17)
  + Dossier: backend/
  + Dockerfile: backend/Dockerfile (runtime eclipse-temurin:17-jre-jammy)
  + Port: 8081 (+ debug 5005)
  + Profil: docker
* **Infra (Docker Compose)**: backend/infra/docker-compose.yml
  + Services: postgres:16.4, mongo:6, minio/minio:latest, quay.io/keycloak/keycloak:24.0.2, backend, frontend
  + Volumes: pgdata, mongodata, miniodata, mavenrepo
* **Config principale (Docker)**: backend/src/main/resources/application-docker.yml
* **Docs API**: Swagger UI via SpringDoc
  + URL: http://localhost:8081/swagger-ui/index.html (après démarrage)

flowchart LR  
 FE[Flutter Web (Nginx:3000)] -->|API REST| BE(Spring Boot:8081)  
 BE -->|JPA| PG[(PostgreSQL)]  
 BE -->|Driver| MG[(MongoDB)]  
 BE -->|S3 API| MN[(MinIO)]  
 BE -->|OIDC| KC[Keycloak]

## 3) Prérequis de développement

* **Java 17** et **Maven >= 3.9**
* **Flutter stable** avec **Dart >= 3** (web activé)
* **Docker Desktop** (Compose v2)
* Accès Internet (dépendances Maven/pub.dev, images Docker de base)

## 4) Démarrage rapide (stack Docker de dev)

1. Créez un fichier .env à la racine de backend/ (référencé par Compose via ../.env depuis backend/infra/). Exemple:

# Base de données  
POSTGRES\_USER=postgres  
POSTGRES\_PASSWORD=postgres  
POSTGRES\_DB=oshapp  
  
# Backend DB  
SPRING\_DATASOURCE\_URL=jdbc:postgresql://postgres:5432/oshapp  
SPRING\_DATASOURCE\_USERNAME=postgres  
SPRING\_DATASOURCE\_PASSWORD=postgres  
  
# Mongo  
MONGO\_INITDB\_DATABASE=oshapp  
  
# MinIO  
MINIO\_ROOT\_USER=minioadmin  
MINIO\_ROOT\_PASSWORD=minioadmin  
  
# Keycloak  
KEYCLOAK\_ADMIN=admin  
KEYCLOAK\_ADMIN\_PASSWORD=admin  
KC\_DB=postgres  
KC\_DB\_URL=jdbc:postgresql://postgres:5432/oshapp  
KC\_DB\_USERNAME=postgres  
KC\_DB\_PASSWORD=postgres  
KEYCLOAK\_AUTH\_SERVER\_URL=http://keycloak:8080  
KEYCLOAK\_CLIENT\_SECRET=CHANGE\_ME  
  
# App  
APP\_JWT\_SECRET=CHANGE\_ME  
APP\_JWT\_EXPIRATIONMS=86400000  
APP\_FRONTEND\_BASE\_URL=oshapp://open  
GOOGLE\_SERVER\_CLIENT\_ID=CHANGE\_ME  
  
# Email  
SPRING\_MAIL\_USERNAME=CHANGE\_ME  
SPRING\_MAIL\_PASSWORD=CHANGE\_ME

1. Démarrez l’infrastructure:

cd backend/infra  
docker compose up -d

1. Accès services par défaut:

* Backend API: http://localhost:8081
* Swagger UI: http://localhost:8081/swagger-ui/index.html
* Keycloak: http://localhost:8080
* MinIO Console: http://localhost:9001
* Frontend (Nginx): http://localhost:3000

Note: le service backend de dev dans Compose utilise l’image maven:3.9-eclipse-temurin-17 et lance spring-boot:run avec le profil docker.

## 5) Variables d’environnement (extrait utile)

Source: backend/src/main/resources/application-docker.yml et backend/infra/docker-compose.yml

* App JWT: APP\_JWT\_SECRET, APP\_JWT\_EXPIRATIONMS
* Frontend deep link/base URL: APP\_FRONTEND\_BASE\_URL
* Google: GOOGLE\_SERVER\_CLIENT\_ID
* Datasource: SPRING\_DATASOURCE\_URL, SPRING\_DATASOURCE\_USERNAME, SPRING\_DATASOURCE\_PASSWORD
* Email: SPRING\_MAIL\_USERNAME, SPRING\_MAIL\_PASSWORD
* Keycloak (app): KEYCLOAK\_AUTH\_SERVER\_URL, KEYCLOAK\_CLIENT\_SECRET, keycloak.realm=oshapp, keycloak.client-id=oshapp-backend
* Keycloak (serveur): KEYCLOAK\_ADMIN, KEYCLOAK\_ADMIN\_PASSWORD, KC\_DB, KC\_DB\_URL, KC\_DB\_USERNAME, KC\_DB\_PASSWORD
* MinIO: MINIO\_ROOT\_USER, MINIO\_ROOT\_PASSWORD

## 6) Dépendances

### 6.1 Frontend (pubspec.yaml — runtime)

* flutter (sdk)
* http: ^1.1.0
* http\_parser: ^4.0.2
* shared\_preferences: ^2.2.2
* flutter\_appauth: ^9.0.1
* url\_launcher: ^6.2.1
* flutter\_secure\_storage: ^9.0.0
* flutter\_typeahead: ^5.2.0
* pin\_code\_fields: 8.0.1
* collection: ^1.18.0
* meta: ^1.15.0
* logger: ^2.0.0
* animated\_background: ^2.0.0
* cupertino\_icons: ^1.0.2
* flutter\_localizations (sdk)
* get\_it: ^7.6.7
* dio: ^5.4.0
* flutter\_svg: ^2.0.10+1
* google\_fonts: ^6.2.1
* font\_awesome\_flutter: ^10.7.0
* google\_sign\_in: 6.2.1
* provider: ^6.1.1
* retrofit: ^4.0.3
* json\_annotation: ^4.8.1
* openid\_client: ^0.4.9
* isar: ^3.1.0+1
* intl: ^0.20.2
* image\_picker: ^1.0.4
* file\_picker: ^8.0.0+1
* csv: ^5.0.2
* excel: ^2.0.4
* path\_provider: ^2.1.1
* flutter\_local\_notifications: ^19.3.1
* firebase\_messaging: ^16.0.0
* fl\_chart: ^1.0.0
* syncfusion\_flutter\_charts: ^30.1.40
* syncfusion\_flutter\_pdfviewer: ^30.1.40
* pdf: ^3.10.7
* printing: ^5.11.1
* qr\_flutter: ^4.1.0
* table\_calendar: ^3.0.9
* timeago: ^3.6.0
* jwt\_decoder: ^2.0.1
* shimmer: ^3.0.0
* google\_nav\_bar: ^5.0.6
* file\_saver: ^0.3.1
* file\_selector: ^1.0.3
* characters: ^1.3.0
* device\_info\_plus: ^11.5.0

### 6.2 Frontend (dev)

* flutter\_lints: ^3.0.0
* build\_runner: ^2.4.8
* json\_serializable: ^6.8.0
* retrofit\_generator: ^8.1.0
* isar\_generator: ^3.1.0+1
* flutter\_gen\_runner: 5.4.0
* flutter\_launcher\_icons: ^0.13.1

### 6.3 Backend (pom.xml)

* spring-boot-starter-web
* spring-boot-starter-validation
* spring-boot-starter-thymeleaf
* spring-boot-starter-mail
* spring-boot-starter-data-jpa
* postgresql (runtime)
* spring-boot-starter-security
* spring-boot-starter-oauth2-resource-server
* spring-boot-devtools (runtime, optional)
* lombok (${lombok.version})
* spring-boot-starter-test (test)
* spring-security-test (test)
* springdoc-openapi-starter-webmvc-ui: 2.5.0
* jjwt-api / jjwt-impl (runtime) / jjwt-jackson: 0.11.5
* javax.annotation-api: 1.3.2
* swagger-annotations: 2.2.20
* mapstruct + mapstruct-processor: ${org.mapstruct.version}
* google-api-client: 2.6.0
* google-http-client-jackson2: 1.44.2

Build plugins clés: maven-compiler-plugin, jacoco-maven-plugin, spring-boot-maven-plugin, maven-surefire-plugin, maven-failsafe-plugin, exec-maven-plugin.

## 7) Build des images applicatives (prod)

Backend:

docker build -t oshapp-backend:0.0.1 -f backend/Dockerfile backend

Frontend (web + Nginx):

docker build -t oshapp-frontend:latest -f frontend/Dockerfile frontend

Optionnel: créez un docker-compose.prod.yml pointant vers ces images au lieu du service backend basé sur Maven.

## 8) Export des images Docker (remise off-line)

# Backend  
docker save oshapp-backend:0.0.1 -o guide/oshapp-backend-0.0.1.tar  
# Frontend  
docker save oshapp-frontend:latest -o guide/oshapp-frontend-latest.tar

## 9) Bases de données — liens et exports

* **PostgreSQL** (Base principale JPA)
  + Service: postgres (5432)
  + Export (depuis Docker):
  + docker exec -t oshapp-postgres pg\_dump -U postgres -d oshapp | gzip > guide/databases/oshapp\_postgres.sql.gz
* **MongoDB** (si utilisé)
  + Service: mongodb (27017)
  + Export:
  + docker exec oshapp-mongo mongodump --db oshapp --archive=guide/databases/oshapp\_mongo.archive
* **MinIO** (stockage objets)
  + Console: http://localhost:9001
  + Export: via mc (MinIO Client) ou téléchargement par la console (à documenter selon bucket utilisé)

Veuillez ajouter ci-dessus les **liens séparés** (drive interne, S3, archive réseau) où sont déposés ces exports.

## 10) Qualité, sécurité et bonnes pratiques

* **JWT**: définir APP\_JWT\_SECRET robuste, conserver hors dépôt.
* **OIDC**: keycloak (realm oshapp, client oshapp-backend). Mettre à jour KEYCLOAK\_CLIENT\_SECRET.
* **Mail**: utiliser un compte SMTP dédié (éviter comptes personnels).
* **Tests & couverture**: jacoco-maven-plugin configuré. Activer mvn test et publier rapports.
* **Mapping**: MapStruct activé (component model Spring).

## 12) Annexes

* Backend application-docker.yml: backend/src/main/resources/application-docker.yml
* Compose: backend/infra/docker-compose.yml
* Dockerfiles: backend/Dockerfile, frontend/Dockerfile
* Ports: Backend 8081 (debug 5005), Keycloak 8080, Postgres 5432, Mongo 27017, MinIO 9000/9001, Frontend 3000