

# TixGo Microservices

## FastAPI-based Microservices Architecture

Sistem microservices berbasis **FastAPI** untuk manajemen **ticketing** dan **event attendance**. Proyek ini mendemonstrasikan implementasi arsitektur microservices dengan:

- **Autentikasi JWT**
- **Otorisasi berbasis role** (committee/admin)
- **Komunikasi inter-service** (Attendance memverifikasi token yang sama)

Repository ini berisi **2 microservice**:

- **identity-service**: authentication & authorization (register, login, me, RBAC)
- **attendance-service**: check-in management (create checkin, fetch attendance per event)

## Overview

TixGo Microservices terdiri dari dua layanan independen yang menangani aspek berbeda dari sistem manajemen event:

### 1. Identity Service (Port 18081)

Menangani:

- Registrasi user baru
- Login & pembuatan token (JWT)
- Verifikasi identitas user
- Manajemen role (committee, admin)
- Endpoint `/auth/me`

### 2. Attendance Service (Port 18082)

Menangani:

- Check-in peserta ke event
- Pencatatan kehadiran per event
- Query data kehadiran (berdasarkan `event_id`)

- Validasi token dari Identity Service (menggunakan `JWT_SECRET` yang sama)

## Arsitektur Sistem

Public Internet / STB

- **Identity Service** (FastAPI) : host port **18081**
- **Attendance Service** (FastAPI) : host port **18082**

Keduanya berjalan di container (Python 3.11), internal container port biasanya `8000/tcp`, dipetakan ke host port `18081/18082`.

## Tech Stack

- FastAPI **0.115.6**
- Python **3.11**
- Docker & Docker Compose
- JWT Authentication (**HS256**)
- In-memory data store (dictionary)

## Deployment Files

- `docker-compose.yml` : development compose
- `docker-compose.prod.yml` : production-ready compose (port host 18081/18082)
- `deploy.sh` : automated deployment untuk STB
- `smoke-test.sh` : quick verification/testing
- `cloudflared/config.yml.example` : contoh Cloudflare Tunnel config
- `.env.example` : template environment variables

## Persyaratan Deployment

### Hardware & Software

- **Docker & Docker Compose**: v2.0+
- **Git**
- **Memory**: minimal 2GB RAM
- **Disk**: minimal 500MB (image + containers)

## Verifikasi Instalasi

```
docker --version
docker-compose --version
git --version
```

## Panduan Deployment di STB

### 1. Step 1 : Clone Repository

```
cd /home/user
git clone https://github.com/ratukhansaaaa/tixgo-microservices.git
cd tixgo-microservices
```

### 2. Step 2 : Setup Environment Variables

**NOTE:** Jangan commit `.env` ke repository. Buat `.env` lokal dengan secrets yang aman.

```
cp .env.example .env
nano .env
```

Isi `.env` (REQUIRED):

```
JWT_SECRET=
JWT_ALG=HS256
TOKEN_EXPIRE_MINUTES=120
```

Generate `JWT_SECRET` yang aman:

```
# OpenSSL
openssl rand -hex 32

# atau Python
python3 -c "import secrets; print(secrets.token_hex(32))"
```

Contoh `.env` valid:

```
JWT_SECRET=a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6
JWT_ALG=HS256
TOKEN_EXPIRE_MINUTES=120
```

### 3. Step 3: Build & Deploy Services

```
docker-compose build
docker-compose -f docker-compose.prod.yml up -d
```

Atau gunakan script:

```
chmod +x deploy.sh
./deploy.sh
```

### 4. Step 4: Verifikasi Deployment

Cek status container:

```
docker ps
```

Expected ports:

- identity → `0.0.0.0:18081->8000/tcp`
- attendance → `0.0.0.0:18082->8000/tcp`

Health check:

```
curl http://localhost:18081/health
curl http://localhost:18082/health
```

## Cara Mengakses Layanan

### 1. Akses Lokal (di STB)

```
# Identity
curl http://localhost:18081/health

# Attendance
curl http://localhost:18082/health
```

## 2. Akses via IP (network yang sama)

```
curl http://<STB_IP>:18081/health
curl http://<STB_IP>:18082/health
```

## 3. Akses via Domain Publik (Recommended)

```
# Identity
curl https://dina.theokaitou.my.id/health
curl https://dina.theokaitou.my.id/auth/login

# Attendance
curl https://ratu.theokaitou.my.id/health
curl https://ratu.theokaitou.my.id/checkins
```

### Domain Mapping:

- `dina.theokaitou.my.id` → `STB_IP:18081` (identity)
- `ratu.theokaitou.my.id` → `STB_IP:18082` (attendance)

## Dokumentasi API

### Identity Service (Port 18081)

#### 1. Health Check

**GET** `/health`

Response `200 OK`:

```
{
  "status": "ok",
  "service": "identity-service"
```

```
}
```

## 2. User Registration

**POST** /auth/register

Auth: tidak perlu

Body:

```
{
  "username": "panitia1",
  "role": "committee"
}
```

Error **400** (contoh):

```
{
  "detail": "User panitia1 sudah terdaftar"
}
```

## 3. Get Attendance by Event

**GET** /attendance/{event\_id}

Auth: **Bearer Token**

Response **200 OK**:

```
{
  "event_id": "evt1",
  "total_checked_in": 2,
  "records": [
    {
      "event_id": "evt1",
      "ticket_id": "TICKET123",
      "checked_in_by": "panitia1",
      "checked_in_at": "2026-01-03T13:25:53.415880+00:00"
    }
  ]
}
```

```
]
}
```

## Autentikasi & Otorisasi

### JWT Token Format

Token `/auth/login` adalah JWT dengan algoritma HS256.

Format: `Header.Payload.Signature`

Payload (decoded) contoh:

```
{
  "sub": "panitia1",
  "role": "committee",
  "iat": 1767446742,
  "exp": 1767453942
}
```

Token Lifetime

- Default: 120 menit (bisa diubah via `TOKEN_EXPIRE_MINUTES`)
- Kalau expired → login ulang

### Authorization Rules

Endpoint	Method	Auth Required	Role Required	Deskripsi
/health (both)	GET	No	-	Health check publik
/auth/register	POST	No	-	Registrasi user
/auth/login	POST	No	-	Login dapat token
/auth/me	GET	Yes	any	Info user sendiri
/checkins	POST	Yes	any	Untuk check-in
/attendance/{id}	GET	Yes	any	Lihat data kehadiran

### Security Notes

- **JWT\_SECRET**: wajib aman, jangan di-commit, dan **harus konsisten** di semua service
- Password di-hash **bcrypt**

- password > 72 bytes akan ditolak (limit bcrypt)
- Gunakan **HTTPS** di production
- Hindari menaruh token di URL atau logs
- Belum ada CORS config (tambahkan bila dibutuhkan)

## Testing

### Automated Smoke Test

```
chmod +x smoke-test.sh
./smoke-test.sh http://localhost:18081 http://localhost:18082
```

Script akan:

1. Check health
2. Register user (jika belum ada)
3. Login & ambil token
4. Create check-in
5. Fetch attendance
6. Report hasil

### Manual Testing Flow (Contoh)

```
# Register
curl -X POST http://localhost:18081/auth/register \
-H "Content-Type: application/json" \
-d '{"username":"testuser","password":"testpass123","role":"committee"}'

# Login -> simpan token
TOKEN=$(curl -s -X POST http://localhost:18081/auth/login \
-H "Content-Type: application/json" \
-d '{"username":"testuser","password":"testpass123"}' | jq -r .access_token)

# Create check-in
curl -X POST http://localhost:18082/checkins \
-H "Content-Type: application/json" \
-H "Authorization: Bearer $TOKEN" \
-d '{"event_id":"evt1","ticket_id":"TKT-001"}'
```



```
# Get attendance
curl -H "Authorization: Bearer $TOKEN" \
http://localhost:18082/attendance/evt1
```

## Troubleshooting

### Containers tidak start

```
docker-compose logs -f
docker-compose down
docker-compose up -d --build
```

### Port 18081 / 18082 sudah dipakai

```
lsof -i :18081
lsof -i :18082
# kill process sesuai PID
kill -9 <PID>
```

Atau ubah port mapping di `docker-compose.yml`.

### JWT\_SECRET error / token invalid, cek .env :

```
cat .env
```

Pastikan:

1. tidak kosong
2. sama persis di semua services
3. minimal 32 karakter

Generate baru:

```
openssl rand -hex 32
docker-compose restart
```

### Token expired

Login ulang :

```
TOKEN=$(curl -s -X POST http://localhost:18081/auth/login \
-H "Content-Type: application/json" \
-d '{"username":"panitia1","password":"secret"}' | jq -r .access_token)
```

### Check-in gagal: "Ticket sudah pernah check-in"

Itu expected behavior (ticket harus unik per event).

Gunakan `ticket_id` baru.

## Project Structure

```
tixgo-microservices/
├── identity-service/
│   ├── app/main.py
│   ├── Dockerfile
│   └── requirements.txt
├── attendance-service/
│   ├── app/main.py
│   ├── Dockerfile
│   └── requirements.txt
├── cloudflared/
│   └── config.yml.example
├── docker-compose.yml
├── docker-compose.prod.yml
├── deploy.sh
├── smoke-test.sh
├── .env          # DO NOT COMMIT
└── README.md
```

## Known Limitations & Future Improvements

- Data masih in-memory → hilang saat container restart
- Belum ada rate limiting / API gateway
- Belum ada monitoring (Prometheus/Grafana)
- Belum ada logging request/response
- Belum ada UI untuk user management
- Belum multi-language

## Development Notes

### Local Development (tanpa Docker)

```
python3 -m venv .venv
source .venv/bin/activate

pip install -r identity-service/requirements.txt
pip install -r attendance-service/requirements.txt

# Terminal 1 - Identity
cd identity-service
JWT_SECRET=dev-secret uvicorn app.main:app --reload --port 8001

# Terminal 2 - Attendance
cd attendance-service
JWT_SECRET=dev-secret uvicorn app.main:app --reload --port 8002
```

### Data Persistence

Sekarang: in-memory (cocok untuk dev/testing).

Untuk production: gunakan DB (PostgreSQL/SQLite), migrations (Alembic), backup strategy.