SharingMezzi

Documentazione API Complete

Reference Completa Endpoint e MQTT

16 giugno 2025

Indice

1 Introduzione

Questa documentazione fornisce una reference completa di tutte le API REST e la comunicazione MQTT del sistema SharingMezzi. Il sistema espone endpoint per gestione utenti, mezzi, corse, parcheggi e amministrazione, oltre a un sistema di comunicazione IoT basato su MQTT.

1.1 Base URL

Sviluppo: `http://localhost:5000/api`
MQTT Broker: `localhost:1883`

1.2 Autenticazione

Il sistema utilizza JWT Bearer tokens per l'autenticazione:

```
Authorization: Bearer {jwt_token}
```

2 Account di Test

Email	Password	Ruolo	Credito
admin@test.com	admin123	Admin	€50.00
mario@test.com	user123	User	€25.00

Tabella 1: Account Predefiniti per Testing

3 API Autenticazione

3.1 POST /api/auth/login

Descrizione: Autenticazione utente e ottenimento JWT token

Request Body:

```
{
    "email": "mario@test.com",
    "password": "user123"
}
```

Response Success (200):

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "user": {
        "id": 2,
        "nome": "Mario",
        "cognome": "Rossi",
        "email": "mario@test.com",
        "credito": 25.00,
        "ruolo": "User",
        "stato": "Attivo"
    }
}
```

Response Error (401):

```
{
    "message": "Email o password non validi"
}
```

Esempio cURL:

```
curl -X POST http://localhost:5000/api/auth/login \
     -H "Content-Type: application/json" \
     -d '{"email":"mario@test.com","password":"user123"}'
```

3.2 POST /api/auth/logout

Descrizione: Invalidazione token JWT

Headers:

```
Authorization: Bearer {token}
```

Response (200):

```
{
    "message": "Logout effettuato con successo"
}
```

4 API Mezzi

4.1 GET /api/mezzi

Descrizione: Lista di tutti i mezzi disponibili

Response (200):

```
[
    {
        "id": 1,
        "modello": "City Bike Classic",
        "tipo": "BiciMuscolare",
        "tariffaPerMinuto": 0.15,
        "tariffaFissa": 1.00,
        "stato": "Disponibile",
        "batteria": null,
        "isElettrico": false,
        "parcheggioAttualeId": 1,
        "nomeParcheggio": "Centro Storico"
    },
    {
        "id": 3,
        "modello": "E-Bike Mountain",
        "tipo": "BiciElettrica",
        "tariffaPerMinuto": 0.30,
        "tariffaFissa": 1.00,
        "stato": "Disponibile",
        "batteria": 85,
        "isElettrico": true,
        "parcheggioAttualeId": 2,
        "nomeParcheggio": "Politecnico"
    }
]
```

4.2 GET /api/mezzi/disponibili

Descrizione: Solo mezzi disponibili per noleggio

Query Parameters:

- `tipo` (optional): Filtra per tipo (BiciMuscolare, BiciElettrica, Monopattino)
- `parcheggioId` (optional): Filtra per parcheggio

Esempio:

```
curl -X GET "http://localhost:5000/api/mezzi/disponibili?tipo=BiciElettrica"
```

4.3 GET /api/mezzi/{id}

Descrizione: Dettagli specifico mezzo

Response (200):

```
{
    "id": 3,
    "modello": "E-Bike Mountain",
    "tipo": "BiciElettrica",
    "tariffaPerMinuto": 0.30,
    "tariffaFissa": 1.00,
    "stato": "Disponibile",
    "batteria": 85,
    "isElettrico": true,
    "parcheggioAttualeId": 2,
    "nomeParcheggio": "Politecnico",
    "createdAt": "2025-06-16T10:00:00Z",
    "ultimaManutenzione": "2025-06-01T09:00:00Z"
}
```

5 API Corse

5.1 POST /api/corse/inizia

Descrizione: Inizia una nuova corsa

Headers:

```
Authorization: Bearer {token}
Content-Type: application/json
```

Request Body:

```
{
    "mezzoId": 3
}
```

Response Success (200):

```
{
    "id": 5,
    "mezzoId": 3,
    "utenteId": 2,
    "stato": "InCorso",
    "inizio": "2025-06-16T14:30:00Z",
    "parcheggioPartenzaId": 2,
```

```
    "nomeParcheggioPartenza": "Politecnico",
    "mezzoModello": "E-Bike Mountain",
    "tariffaPerMinuto": 0.30,
    "tariffaFissa": 1.00
}
```

Response Error (400):

```
{
    "message": "Mezzo non disponibile o gi  in uso"
}
```

5.2 PUT /api/corse/{id}/termina

Descrizione: Termina corsa esistente

Request Body:

```
{
    "parcheggioDestinazioneId": 3,
    "segnalazioneManutenzione": "Freni rumorosi e batteria problematica"
}
```

Response Success (200):

```
{
    "id": 5,
    "inizio": "2025-06-16T14:30:00Z",
    "fine": "2025-06-16T15:15:00Z",
    "durataMinuti": 45,
    "costoTotale": 14.50,
    "stato": "Completata",
    "parcheggioDestinazioneId": 3,
    "nomeParcheggioDestinazione": "Porta Nuova",
    "segnalazioneManutenzioneCreata": true
}
```

5.3 GET /api/corse/attive

Descrizione: Corse attualmente in corso per l'utente

Headers:

```
Authorization: Bearer {token}
```

5.4 GET /api/corse/storico

Descrizione: Storico corse dell'utente

Query Parameters:

- `limit` (default: 50): Numero massimo risultati
- `offset` (default: 0): Offset per paginazione

6 API Parcheggi

6.1 GET /api/parcheggi

Descrizione: Lista di tutti i parcheggi

Response (200):

```
[
    {
        "id": 1,
        "nome": "Centro Storico",
        "indirizzo": "Piazza Castello 1, Torino",
        "latitude": 45.0703,
        "longitude": 7.6869,
        "capacita": 25,
        "postiLiberi": 23,
        "mezziPresenti": 2,
        "mezzi": [
            {
                "id": 1,
                "modello": "City Bike Classic",
                "tipo": "BiciMuscolare",
                "stato": "Disponibile"
            }
        ]
    }
]
```

6.2 GET /api/parcheggi/{id}

Descrizione: Dettagli specifico parcheggio con mezzi presenti

7 API Utente

7.1 GET /api/user/profile

Descrizione: Profilo utente corrente

Headers:

```
Authorization: Bearer {token}
```

Response (200):

```
{
    "id": 2,
    "nome": "Mario",
    "cognome": "Rossi",
    "email": "mario@test.com",
    "telefono": "+39 123 456 7890",
    "credito": 25.00,
    "ruolo": "User",
    "stato": "Attivo",
    "dataRegistrazione": "2025-01-15T10:00:00Z",
    "ultimoAccesso": "2025-06-16T14:30:00Z",
    "totaleCorse": 12,
    "totaleSpeso": 156.75
}
```

7.2 POST /api/user/ricarica-credito

Descrizione: Ricarica credito utente

Request Body:

```
{
    "utenteId": 2,
    "importo": 10.00,
    "metodoPagamento": "CartaCredito"
}
```

Response Success (200):

```
{
    "success": true,
    "nuovoCredito": 35.00,
    "transazioneId": "TXN_20250616_001",
    "message": "Ricarica di   10  .00 completata con successo"
}
```

7.3 PUT /api/user/profile

Descrizione: Aggiorna profilo utente

Request Body:

```
{
    "nome": "Mario",
    "cognome": "Rossi",
    "telefono": "+39 123 456 7890"
}
```

8 API Amministrazione

8.1 GET /api/admin/system-status

Descrizione: Stato generale del sistema

Headers:

```
Authorization: Bearer {admin_token}
```

Response (200):

```
{
    "timestamp": "2025-06-16T14:30:00Z",
    "totalMezzi": 6,
    "mezziDisponibili": 4,
    "mezziInUso": 1,
    "mezziManutenzione": 1,
    "totaleParcheggi": 3,
    "ioTDevicesConnected": 6,
    "corsaAttive": 1,
    "batterieBasse": 0,
    "segnalazioniAperte": 2,
    "sistemaStatus": "Operativo"
}
```

8.2 GET /api/admin/mezzi

Descrizione: Lista completa mezzi con dettagli admin
Response (200):

```
[
    {
        "id": 3,
        "modello": "E-Bike Mountain",
        "tipo": "BiciElettrica",
        "stato": "Manutenzione",
        "batteria": 45,
        "parcheggioAttualeId": 2,
        "ultimaManutenzione": "2025-06-01T09:00:00Z",
        "chilometraggio": 1250.5,
        "numeroCorse": 78,
        "segnalazioniAperte": 1,
        "costoManutenzione": 45.00
    }
]
```

8.3 POST /api/admin/schedule-maintenance/{mezzoId}

Descrizione: Programma manutenzione per un mezzo
Request Body:

```
{
    "note": "Manutenzione programmata - controllo freni",
    "priorita": "Media",
    "dataScadenza": "2025-06-20T09:00:00Z"
}
```

8.4 GET /api/admin/segnalazioni

Descrizione: Lista segnalazioni manutenzione
Query Parameters:

- `stato`: Filtra per stato (Aperta, InCorso, Completata)
- `priorita`: Filtra per priorità (Bassa, Media, Alta, Urgente)

8.5 PUT /api/admin/segnalazioni/{id}/stato

Descrizione: Aggiorna stato segnalazione
Request Body:

```
{
    "nuovoStato": "InCorso",
    "note": "Manutenzione iniziata - tecnico Mario"
}
```

9 API Testing e Diagnostica

9.1 GET /api/test/mqtt

Descrizione: Test connessione MQTT broker

Response (200):

```
{
    "mqttBrokerStatus": "Connected",
    "connectedClients": 6,
    "messagesReceived": 1245,
    "messagesSent": 892,
    "uptime": "02:15:30"
}
```

9.2 POST /api/test/iot-command

Descrizione: Invia comando test a dispositivo IoT
Request Body:

```
{
    "mezzoId": "BIC001",
    "command": "battery_check",
    "parameters": {}
}
```

10 Comunicazione MQTT

10.1 Configurazione Broker

Host: localhost
Port: 1883
Protocol: MQTT 3.1.1
QoS: 1 (At least once delivery)

10.2 Topic Structure

10.2.1 Device Status Topics

```
sharingmezzi/devices/{mezzoId}/status
sharingmezzi/devices/{mezzoId}/battery
sharingmezzi/devices/{mezzoId}/location
sharingmezzi/devices/{mezzoId}/diagnostics
```

10.2.2 Command Topics

```
sharingmezzi/commands/{mezzoId}/unlock
sharingmezzi/commands/{mezzoId}/lock
sharingmezzi/commands/{mezzoId}/maintenance
sharingmezzi/commands/{mezzoId}/reset
```

10.2.3 Notification Topics

```
sharingmezzi/notifications/battery_low
sharingmezzi/notifications/maintenance_required
sharingmezzi/notifications/device_offline
sharingmezzi/notifications/emergency
```

10.3 Message Formats

10.3.1 Battery Update

Topic: sharingmezzi/devices/BIC001/battery

```
{
    "mezzoId": "BIC001",
    "batteryLevel": 85,
    "voltage": 12.4,
    "temperature": 25.5,
    "chargingStatus": "not_charging",
    "estimatedRange": 25,
    "timestamp": "2025-06-16T14:30:00Z"
}
```

10.3.2 Location Update

Topic: sharingmezzi/devices/SCO004/location

```
{
    "mezzoId": "SCO004",
    "latitude": 45.0703,
    "longitude": 7.6869,
    "accuracy": 5.0,
    "speed": 12.5,
    "heading": 45.2,
    "timestamp": "2025-06-16T14:30:00Z"
}
```

10.3.3 Status Update

Topic: sharingmezzi/devices/BIC002/status

```
{
    "mezzoId": "BIC002",
    "status": "in_use",
    "userId": "user123",
    "sessionId": "SESS_20250616_001",
    "locked": false,
    "maintenanceMode": false,
    "lastActivity": "2025-06-16T14:30:00Z",
    "timestamp": "2025-06-16T14:30:00Z"
}
```

10.3.4 Unlock Command

Topic: sharingmezzi/commands/BIC001/unlock

```
{
    "command": "unlock",
    "mezzoId": "BIC001",
    "userId": "user123",
    "sessionId": "SESS_20250616_001",
    "duration": 7200,
    "timestamp": "2025-06-16T14:30:00Z",
    "signature": "sha256_hash_for_security"
}
```

10.3.5 Maintenance Notification

Topic: `sharingmezzi/notifications/maintenance_required`

```
{
    "mezzoId": "BIC003",
    "alertType": "maintenance_required",
    "priority": "high",
    "description": "Freni rumorosi segnalati da utente",
    "reportedBy": "user123",
    "timestamp": "2025-06-16T14:30:00Z",
    "autoGenerated": false
}
```

10.4 Client Connection Examples

11 Codici di Errore

Codice	Descrizione	Azione Suggerita
200	OK	Richiesta completata con successo
201	Created	Risorsa creata con successo
400	Bad Request	Controllare parametri richiesta
401	Unauthorized	Token JWT non valido o scaduto
403	Forbidden	Permissions insufficienti
404	Not Found	Risorsa non trovata
409	Conflict	Conflitto stato risorsa (es. mezzo già in uso)
422	Unprocessable Entity	Errori validazione dati
500	Internal Server Error	Errore interno server
503	Service Unavailable	Servizio temporaneamente non disponibile

12 Esempi Workflow Completi

12.1 Workflow Noleggio Completo

1. Login Utente:

```
curl -X POST http://localhost:5000/api/auth/login \
     -H "Content-Type: application/json" \
     -d '{"email":"mario@test.com","password":"user123"}'
```

2. Lista Mezzi Disponibili:

```
curl -X GET http://localhost:5000/api/mezzi/disponibili \
     -H "Authorization: Bearer {token}"
```

3. Inizia Corsa:

```
curl -X POST http://localhost:5000/api/corse/inizia \
     -H "Authorization: Bearer {token}" \
     -H "Content-Type: application/json" \
     -d '{"mezzoId": 3}'
```

4. Termina Corsa con Segnalazione:

```
curl -X PUT http://localhost:5000/api/corse/5/termina \
     -H "Authorization: Bearer {token}" \
     -H "Content-Type: application/json" \
     -d '{
       "parcheggioDestinazioneId": 3,
       "segnalazioneManutenzione": "Freni rumorosi"
     }'
```

12.2 Workflow Amministrazione

1. Login Admin:

```
curl -X POST http://localhost:5000/api/auth/login \
     -H "Content-Type: application/json" \
     -d '{"email":"admin@test.com","password":"admin123"}'
```

2. Stato Sistema:

```
curl -X GET http://localhost:5000/api/admin/system-status \
     -H "Authorization: Bearer {admin_token}"
```

3. Programma Manutenzione:

```
curl -X POST http://localhost:5000/api/admin/schedule-maintenance/3 \
     -H "Authorization: Bearer {admin_token}" \
     -H "Content-Type: application/json" \
     -d '{"note": "Controllo freni programmato"}'
```

13 Monitoraggio e Logging

13.1 Health Check Endpoints

- `GET /health` - Stato generale applicazione
- `GET /health/database` - Connessione database
- `GET /health/mqtt` - Stato broker MQTT
- `GET /health/signalr` - Hubs SignalR