Merendels Backend - Guida Test API

Base URL: `http://localhost:8080`

---

1. HEALTH CHECK

Verifica stato server

```bash
curl http://localhost:8080/health
```

Response:

```json
{
  "status": "OK",
  "message": "Merendels Backend is running",
  "version": "0.0.1"
}
```

---

2. AUTENTICAZIONE

2.1 Registrazione nuovo utente

```bash
curl -X POST http://localhost:8080/api/auth/register \
  -H "Content-Type: application/json" \
  -d '{
    "name": "Mario Rossi",
    "email": "mario.rossi@test.com",
    "password": "password123"
  }'
```

2.2 Login

```bash
```

```bash
curl -X POST http://localhost:8080/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{
    "email": "mario.rossi@test.com",
    "password": "password123"
  }'
```

Response esempio:

```json
{
  "message": "Login successful",
  "data": {
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "user": {
      "id": 1,
      "name": "Mario Rossi",
      "email": "mario.rossi@test.com"
    }
  }
}
```

⚠️ **IMPORTANTE:** Salva il `token` per le chiamate successive!

2.3 Profilo utente (richiede token)

```bash
curl -X GET http://localhost:8080/api/auth/profile \
  -H "Authorization: Bearer TOKEN_QUI"
```

---

3. GESTIONE RUOLI UTENTE

3.1 Lista tutti i ruoli

```bash
curl -X GET http://localhost:8080/api/user-roles \
  -H "Authorization: Bearer TOKEN_QUI"
```

3.2 Singolo ruolo per ID

```bash
```

```
curl -X GET http://localhost:8080/api/user-roles/1 \
  -H "Authorization: Bearer TOKEN_QUI"
```

3.3 Creare nuovo ruolo (solo admin/responsabile)

bash

```
curl -X POST http://localhost:8080/api/user-roles \
  -H "Authorization: Bearer TOKEN_QUI" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "Manager",
    "hierarchy_level": 2
  }'
```

---

4. TIMBRATURE (Funzionalità Principale)

4.1 Creare nuova timbratura

bash

```
curl -X POST http://localhost:8080/api/timbrature \
  -H "Authorization: Bearer TOKEN_QUI" \
  -H "Content-Type: application/json" \
  -d '{
    "action_type": "ENTRATA",
    "location": "UFFICIO"
  }'
```

Valori possibili:

- `action_type`: `"ENTRATA"` o `"USCITA"`
- `location`: `"UFFICIO"` o `"SMART"`

4.2 Le mie timbrature

bash

```
curl -X GET http://localhost:8080/api/timbrature/me \
  -H "Authorization: Bearer TOKEN_QUI"
```

4.3 Timbrature di oggi

bash

```
curl -X GET http://localhost:8080/api/timbrature/me/today \
  -H "Authorization: Bearer TOKEN_QUI"
```

4.4 Timbrature per data specifica

bash

```
curl -X GET http://localhost:8080/api/timbrature/me/date/2025-09-03 \
  -H "Authorization: Bearer TOKEN_QUI"
```

4.5 Stato lavorativo corrente

bash

```
curl -X GET http://localhost:8080/api/timbrature/me/status \
  -H "Authorization: Bearer TOKEN_QUI"
```

4.6 Ultima timbratura

bash

```
curl -X GET http://localhost:8080/api/timbrature/me/last \
  -H "Authorization: Bearer TOKEN_QUI"
```

4.7 Tutte le timbrature (solo admin/responsabile)

bash

```
curl -X GET http://localhost:8080/api/timbrature \
  -H "Authorization: Bearer TOKEN_QUI"
```

---

5. GESTIONE RICHIESTE FERIE/PERMESSI

5.1 Creare richiesta ferie

bash

```
curl -X POST http://localhost:8080/api/requests \
  -H "Authorization: Bearer TOKEN_QUI" \
  -H "Content-Type: application/json" \
  -d '{
    "start_date": "2025-12-20",
    "end_date": "2025-12-27",
    "request_type": "FERIE",
    "notes": "Vacanze natalizie"
  }'
```

Valori possibili:

- `request_type`: `"FERIE"` o `"PERMESSO"`

5.2 Le mie richieste

bash

```
curl -X GET http://localhost:8080/api/requests/me \
  -H "Authorization: Bearer TOKEN_QUI"
```

5.3 Tutte le richieste (solo admin/responsabile)

bash

```
curl -X GET http://localhost:8080/api/requests \
  -H "Authorization: Bearer TOKEN_QUI"
```

---

6. APPROVAZIONI

6.1 Creare approvazione (solo admin/responsabile)

bash

```
curl -X POST http://localhost:8080/api/approvals \
  -H "Authorization: Bearer TOKEN_QUI" \
  -H "Content-Type: application/json" \
  -d '{
    "request_id": 1,
    "status": "APPROVED",
    "comments": "Richiesta approvata"
  }'
```

Valori possibili per status:

- `"APPROVED"` (approvata)
- `"REJECTED"` (rifiutata)
- `"REVOKED"` (revocata)

6.2 Le mie approvazioni

```bash
curl -X GET http://localhost:8080/api/approvals/me \
  -H "Authorization: Bearer TOKEN_QUI"
```

---

7. SEQUENZA TEST COMPLETA

Step 1: Verifica server

```bash
curl http://localhost:8080/health
```

Step 2: Registrati

```bash
curl -X POST http://localhost:8080/api/auth/register \
  -H "Content-Type: application/json" \
  -d '{"name":"Test User","email":"test@demo.com","password":"test123"}'
```

Step 3: Login (salva il token dalla response!)

```bash
curl -X POST http://localhost:8080/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{"email":"test@demo.com","password":"test123"}'
```

Step 4: Prima timbratura (ENTRATA)

```bash
curl -X POST http://localhost:8080/api/timbrature \
  -H "Authorization: Bearer TOKEN_DELLA_STEP_3" \
  -H "Content-Type: application/json" \
  -d '{"action_type":"ENTRATA","location":"UFFICIO"}'
```

Step 5: Seconda timbratura (USCITA)

```bash
curl -X POST http://localhost:8080/api/timbrature \
  -H "Authorization: Bearer TOKEN_DELLA_STEP_3" \
  -H "Content-Type: application/json" \
  -d '{"action_type":"USCITA","location":"UFFICIO"}'
```

Step 6: Verifica timbrature

```bash
curl -X GET http://localhost:8080/api/timbrature/me \
  -H "Authorization: Bearer TOKEN_DELLA_STEP_3"
```

---

8. UTENTI PRE-CREATI (dal database)

Per test immediati, nel database sono già presenti questi utenti:

Admin/Capo:

- **Email:** `giacomo.festuccia@merendels.it`
- **Password:** `password123` (da testare)
- **Ruolo:** Capo (hierarchy_level: 1)

Dipendente:

- **Email:** `test@example.com`
- **Password:** `password123` (da testare)
- **Ruolo:** Dipendente

---

9. ERRORI COMUNI

401 Unauthorized

```json
{"error": "Invalid or expired token"}
```

Soluzione: Verifica che il token sia valido e incluso nell'header Authorization.

400 Bad Request

```
json

{"error": "Invalid request format"}
```

Soluzione: Controlla il formato JSON nel body della richiesta.

409 Conflict (Timbrature)

```
json

{"error": "You already entered today. You must exit first"}
```

Soluzione: Rispetta la sequenza ENTRATA → USCITA → ENTRATA...

---

10. NOTE TECNICHE

- **Base URL:** `http://localhost:8080`
- **Autenticazione:** JWT Bearer Token
- **Content-Type:** `application/json` per POST/PUT
- **CORS:** Configurato per accettare richieste cross-origin
- **Database:** PostgreSQL con dati di esempio precaricati
- **Timezone:** UTC per tutti i timestamp

Per test rapidi, usa un tool come Postman oppure i comandi curl sopra!