# Manual API — Auth, Device, Profile (NestJS)

**Base URL (dev):** http://localhost:3001

**Autentificare:** - JwtGuard = necesită Authorization: Bearer <JWT> - AdminGuard = rol/permisiune administrativă (implică autentificare)

**Conținut:** application/json

## 1) AUTH

@Controller('auth')

### 1.1 Signup

**Path:** /auth/signup  
**Method:** POST  
**Summary:** Creează cont nou utilizator.  
**Security:** Public

**Body (SignupDto):**

{  
 "username": "string|min:4",  
 "email": "email",  
 "password": "string|min:6",  
 "role": "Role(enum)"  
}

**Responses:** - **201 Created** – utilizator creat (ex. obiect user/token, depinde de service) - **400 Bad Request** – validare eșuată - **409 Conflict** – email/username existent

### 1.2 Login

**Path:** /auth/login  
**Method:** POST  
**Summary:** Autentifică și returnează token.  
**Security:** Public

**Body (LoginDto):**

{  
 "identifier": "string (email sau username)",  
 "password": "string"  
}

**Responses:** - **200 OK** json { "accessToken": "<jwt>", "refreshToken": "<jwt>", "expiresIn": 3600, "tokenType": "Bearer" } - **401 Unauthorized** – credențiale greșite - **400 Bad Request** – validare

### 1.3 Logout

**Path:** /auth/logout  
**Method:** POST  
**Summary:** Invalidează sesiunea/refresh token pentru utilizator.  
**Security:** Public (în cod nu e protejat; recomandat JwtGuard în producție)

**Body:**

{ "userId": 123 }

**Responses:** - **200 OK** – sesiune invalidată - **404 Not Found** – utilizator inexistent

### 1.4 Refresh token

**Path:** /auth/refresh-token  
**Method:** POST  
**Summary:** Emite un nou access token pe baza refresh token-ului.  
**Security:** Public (validarea se face pe token-ul trimis)

**Body:**

{ "refreshToken": "<jwt>" }

**Responses:** - **200 OK** – token nou json { "accessToken": "<jwt>", "expiresIn": 3600, "tokenType": "Bearer" } - **403 Forbidden** – Invalid refresh token

### 1.5 Verify email

**Path:** /auth/verify-email  
**Method:** GET  
**Summary:** Confirmă adresa de email pe baza token-ului.  
**Security:** Public

**Query params:** - token (string, required)

**Responses:** - **200 OK** – email verificat - **400/403** – token invalid/expirat

### 1.6 Admin — Update user

**Path:** /auth/admin/:id  
**Method:** PUT  
**Summary:** Actualizează câmpuri de autentificare pentru un utilizator.  
**Security:** AdminGuard

**Path params:** - id (number, required)

**Body (AdminUpdateUserDto):** (toate opționale)

{  
 "username": "string|min:4",  
 "email": "email",  
 "role": "Role(enum)",  
 "isBanned": true  
}

**Responses:** - **200 OK** – utilizator actualizat - **404 Not Found** – utilizator inexistent - **403 Forbidden** – nu ai drepturi

### 1.7 Admin — Get all users

**Path:** /auth/admin  
**Method:** GET  
**Summary:** Listează toți utilizatorii.  
**Security:** AdminGuard

**Responses:** - **200 OK** – User[] - **403 Forbidden**

### 1.8 Admin — Get user by id

**Path:** /auth/admin/:id  
**Method:** GET  
**Summary:** Obține utilizator după id.  
**Security:** AdminGuard

**Path params:** id (number)

**Responses:** - **200 OK** – User - \*\*404 Not Found- \*\*403 Forbidden

## 2) DEVICE

@UseGuards(JwtGuard) + @Controller('device')

### 2.1 Create device

**Path:** /device/create  
**Method:** POST  
**Summary:** Creează un dispozitiv pentru utilizatorul curent.  
**Security:** JwtGuard

**Body (CreateDeviceDto):**

{  
 "name": "string|max:20",  
 "maxConsumption": 100,  
 "room": "Room(enum)",  
 "status": "Status(enum, optional)",  
 "type": "Type(enum)"  
}

**Responses:** - **201 Created** – Device - **400 Bad Request** – validare

### 2.2 Get my devices

**Path:** /device/my-devices  
**Method:** GET  
**Summary:** Listează dispozitivele utilizatorului curent.  
**Security:** JwtGuard

**Responses:** - **200 OK** – Device[]

### 2.3 Update my device by id

**Path:** /device/update-device/:id  
**Method:** PUT  
**Summary:** Actualizează dispozitivul propriu după id.  
**Security:** JwtGuard

**Path params:** id (number)

**Body (UpdateDeviceDto — toate opționale):**

{  
 "name": "string|max:20",  
 "maxConsumption": 120,  
 "room": "Room(enum)",  
 "status": "Status(enum)",  
 "type": "Type(enum)"  
}

**Responses:** - **200 OK** – Device - **404 Not Found** – nu există sau nu aparține utilizatorului

### 2.4 Delete my device by id

**Path:** /device/delete-device/:id  
**Method:** DELETE  
**Summary:** Șterge dispozitivul propriu după id.  
**Security:** JwtGuard

**Path params:** id (number)

**Responses:** - **200 OK** – confirmare ștergere / entitatea ștearsă - **404 Not Found**

### 2.5 Admin — Get all devices

**Path:** /device/admin  
**Method:** GET  
**Summary:** Listează toate dispozitivele.  
**Security:** AdminGuard

**Responses:** - **200 OK** – Device[] - \*\*403 Forbidden`

### 2.6 Admin — Get device by id

**Path:** /device/admin/:id  
**Method:** GET  
**Summary:** Obține dispozitiv după id.  
**Security:** AdminGuard

**Path params:** id (number)

**Responses:** - **200 OK** – Device - \*\*404 Not Found`

### 2.7 Admin — Update device by id

**Path:** /device/admin/:id  
**Method:** PUT  
**Summary:** Actualizează dispozitiv după id.  
**Security:** AdminGuard

**Path params:** id (number)

**Body:** UpdateDeviceDto (vezi 2.3)

**Responses:** - **200 OK** – Device - \*\*404 Not Found`

### 2.8 Admin — Delete device by id

**Path:** /device/admin/:id  
**Method:** DELETE  
**Summary:** Șterge dispozitiv după id.  
**Security:** AdminGuard

**Path params:** id (number)

**Responses:** - **200 OK** – confirmare - \*\*404 Not Found`

## 3) PROFILE

@UseGuards(JwtGuard) + @Controller('profile')

### 3.1 Create profile

**Path:** /profile/create  
**Method:** POST  
**Summary:** Creează profil pentru utilizatorul curent.  
**Security:** JwtGuard

**Body (CreateProfileDto):**

{  
 "firstName": "string|max:20",  
 "lastName": "string|max:20",  
 "description": "string|max:200",  
 "avatarImage": "string (optional)"  
}

**Responses:** - **201 Created** – Profile - **400 Bad Request** – validare

### 3.2 Get my profile

**Path:** /profile/me  
**Method:** GET  
**Summary:** Returnează profilul utilizatorului curent.  
**Security:** JwtGuard

**Responses:** - **200 OK** – Profile - **404 Not Found** – profil inexistent

### 3.3 Update my profile

**Path:** /profile/update  
**Method:** PUT  
**Summary:** Actualizează profilul utilizatorului curent.  
**Security:** JwtGuard

**Body (UpdateProfileDto — toate opționale):**

{  
 "firstName": "string|max:20",  
 "lastName": "string|max:20",  
 "description": "string|max:200",  
 "avatarImage": "string"  
}

**Responses:** - **200 OK** – Profile - **404 Not Found**

### 3.4 Admin — Get all profiles

**Path:** /profile/admin  
**Method:** GET  
**Summary:** Listează toate profilele.  
**Security:** AdminGuard

**Responses:** - **200 OK** – Profile[] - \*\*403 Forbidden`

### 3.5 Admin — Get profile by user id

**Path:** /profile/admin/:id  
**Method:** GET  
**Summary:** Obține profil după userId.  
**Security:** AdminGuard

**Path params:** id (number)

**Responses:** - **200 OK** – Profile - \*\*404 Not Found`

### 3.6 Admin — Update profile by user id

**Path:** /profile/admin/:id  
**Method:** PUT  
**Summary:** Actualizează profil după userId.  
**Security:** AdminGuard

**Path params:** id (number)

**Body:** UpdateProfileDto (vezi 3.3)

**Responses:** - **200 OK** – Profile - \*\*404 Not Found`