📘 Spécification de l’API: Peekaboo

> **Version :** 1.0.0
> **Description :** API pour gérer les oiseaux, leurs historiques de localisation et l’authentification des utilisateurs.
> **Serveur :** `http://localhost:8010`

🔗 `/birds`

Méthode	Description	Réponses HTTP
GET	Récupère la liste de tous les oiseaux.	`200` - Opération réussie

🔗 `/user/{userId}/birds`

Méthode	Description	Réponses HTTP
GET	Récupère tous les oiseaux appartenant à un utilisateur donné	`200` - Liste d’oiseaux `404` - Utilisateur non trouvé

Paramètre de chemin :

Nom	Type	Obligatoire	Description
userId	integer	✅	ID de l’utilisateur

🔗 `/bird/{gpsId}/locations`

Méthode	Description	Réponses HTTP
POST	Met à jour les positions d’un oiseau via son GPS ID	`200` - Mise à jour réussie `400` - Requête invalide `404` - Oiseau non trouvé

Paramètre de chemin :

Nom	Type	Obligatoire	Description
gpsId	string	✅	Identifiant GPS de l’oiseau

Corps de la requête :

- Un objet contenant un tableau de mises à jour de localisation (`locations[]`)

🔗 `/bird/{id}/location`

Méthode	Description	Réponses HTTP
GET	Récupère la dernière position connue d’un oiseau	`200` - Dernière position `404` - Oiseau ou historique de localisation introuvable

Paramètre de chemin :

Nom	Type	Obligatoire	Description
id	integer	✅	ID de l’oiseau

🔗 `/bird/{id}/path`

Méthode	Description	Réponses HTTP
GET	Récupère l’historique complet des positions de l’oiseau donné	`200` - Historique récupéré `404` - Oiseau non trouvé

Paramètre de chemin :

Nom	Type	Obligatoire	Description
id	integer	✅	ID de l’oiseau

🔗 `/login`

Méthode	Description	Réponses HTTP
POST	Authentifie un utilisateur et retourne un JWT	`200` - Succès (token JWT) `401` - Identifiants invalides

Corps de la requête :

- email (string)
- password (string)

🔗 `/register`

Méthode	Description	Réponses HTTP
POST	Enregistre un nouvel utilisateur	`201` - Enregistrement réussi `400` - Données manquantes

Corps de la requête :

- email (string)
- password (string)

Schéma :

```yaml
paths:
  /birds:
    get:
      summary: Get All Birds
      description: Retrieves a list of all birds.
      responses:
        "200":
          description: Successful operation
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/Bird'

  /user/{userId}/birds:
    get:
      summary: Get Birds for a Specific User
      description: Retrieves all birds owned by a given user.
      parameters:
        - name: userId
          in: path
          required: true
          schema:
            type: integer
      responses:
        "200":
          description: List of birds for the user.
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/Bird'
        "404":
          description: User not found.
          content:
            application/json:
              schema:
                type: object
                properties:
                  error:
                    type: string
                    example: "User not found"

  /bird/{gpsId}/locations:
    post:
      summary: Update Bird Locations
      description: Updates the location of a bird identified by its GPS ID.
      parameters:
        - name: gpsId
```

```
          in: path
          required: true
          schema:
            type: string
      requestBody:
        description: An object containing an array of location updates.
        required: true
        content:
          application/json:
            schema:
              type: object
              required:
                - locations
              properties:
                locations:
                  type: array
                  items:
                    $ref: '#/components/schemas/LocationUpdate'
      responses:
        "200":
          description: Locations updated successfully.
          content:
            application/json:
              schema:
                type: object
                properties:
                  message:
                    type: string
                    example: "Locations updated successfully"
        "400":
          description: Invalid payload.
          content:
            application/json:
              schema:
                type: object
                properties:
                  error:
                    type: string
                    example: "Invalid payload"
        "404":
          description: Bird not found.
          content:
            application/json:
              schema:
                type: object
                properties:
                  error:
                    type: string
                    example: "Bird not found"

  /bird/{id}/location:
    get:
      summary: Get Last Location of a Bird
      description: Retrieves the most recent location for the specified bird.
      parameters:
        - name: id
          in: path
          required: true
          schema:
            type: integer
      responses:
        "200":
          description: Last location found.
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Location'
        "404":
          description: Bird or location history not found.
          content:
            application/json:
              schema:
                type: object
                properties:
                  error:
                    type: string
                    example: "Bird not found"  # or "No location history four

  /bird/{id}/path:
    get:
      summary: Get Bird's Path (Location History)
      description: Retrieves the complete ordered location history for a spec
      parameters:
        - name: id
          in: path
          required: true
          schema:
            type: integer
      responses:
        "200":
          description: Location history retrieved successfully.
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/Location'
        "404":
          description: Bird not found.
          content:
            application/json:
              schema:
                type: object
                properties:
```

```yaml
                  error:
                    type: string
                    example: "Bird not found"

  /login:
    post:
      summary: User Login
      description: Authenticates a user and returns a JWT token.
      requestBody:
        description: User credentials.
        required: true
        content:
          application/json:
            schema:
              type: object
              required:
                - email
                - password
              properties:
                email:
                  type: string
                  example: "user@example.com"
                password:
                  type: string
                  example: "yourpassword"
      responses:
        "200":
          description: Login successful.
          content:
            application/json:
              schema:
                type: object
                properties:
                  token:
                    type: string
                    example: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
        "401":
          description: Invalid credentials.
          content:
            application/json:
              schema:
                type: object
                properties:
                  error:
                    type: string
                    example: "Invalid credentials"

  /register:
    post:
      summary: User Registration
      description: Registers a new user.
      requestBody:
        description: User registration data.
        required: true
        content:
          application/json:
            schema:
              type: object
              required:
                - email
                - password
              properties:
                email:
                  type: string
                  example: "user@example.com"
                password:
                  type: string
                  example: "yourpassword"
      responses:
        "201":
          description: User registered successfully.
          content:
            application/json:
              schema:
                type: object
                properties:
                  message:
                    type: string
                    example: "User registered successfully"
        "400":
          description: Missing email or password.
          content:
            application/json:
              schema:
                type: object
                properties:
                  error:
                    type: string
                    example: "Email and password are required"

components:
  schemas:
    Bird:
      type: object
      properties:
        id:
          type: integer
          example: 1
        name:
          type: string
          example: "Sparrow"
        latitude:
          type: number
          format: float
          example: 40.7128
        longitude:
          type: number
          format: float
          example: -74.0060
        owner:
          type: string
          description: Username of the bird owner.
          example: "user@example.com"
        gps_id:
          type: string
          example: "ABC123"
    Location:
      type: object
      properties:
        latitude:
          type: number
          format: float
          example: 40.7128
        longitude:
          type: number
          format: float
          example: -74.0060
        timestamp:
          type: string
          format: date-time
          example: "2025-03-31 12:34:56"
    LocationUpdate:
      type: object
      properties:
        latitude:
          type: number
          format: float
          example: 40.7128
        longitude:
          type: number
          format: float
          example: -74.0060
    User:
      type: object
      properties:
        id:
          type: integer
          example: 1
        email:
          type: string
          example: "user@example.com"
        password:
          type: string
          description: Hashed password.
          example: "$2y$10$..."
        roles:
```

```yaml
type: array
items:
  type: string
example:
  - "ROLE_USER"
```